Group Communication (Application-level Multicast)

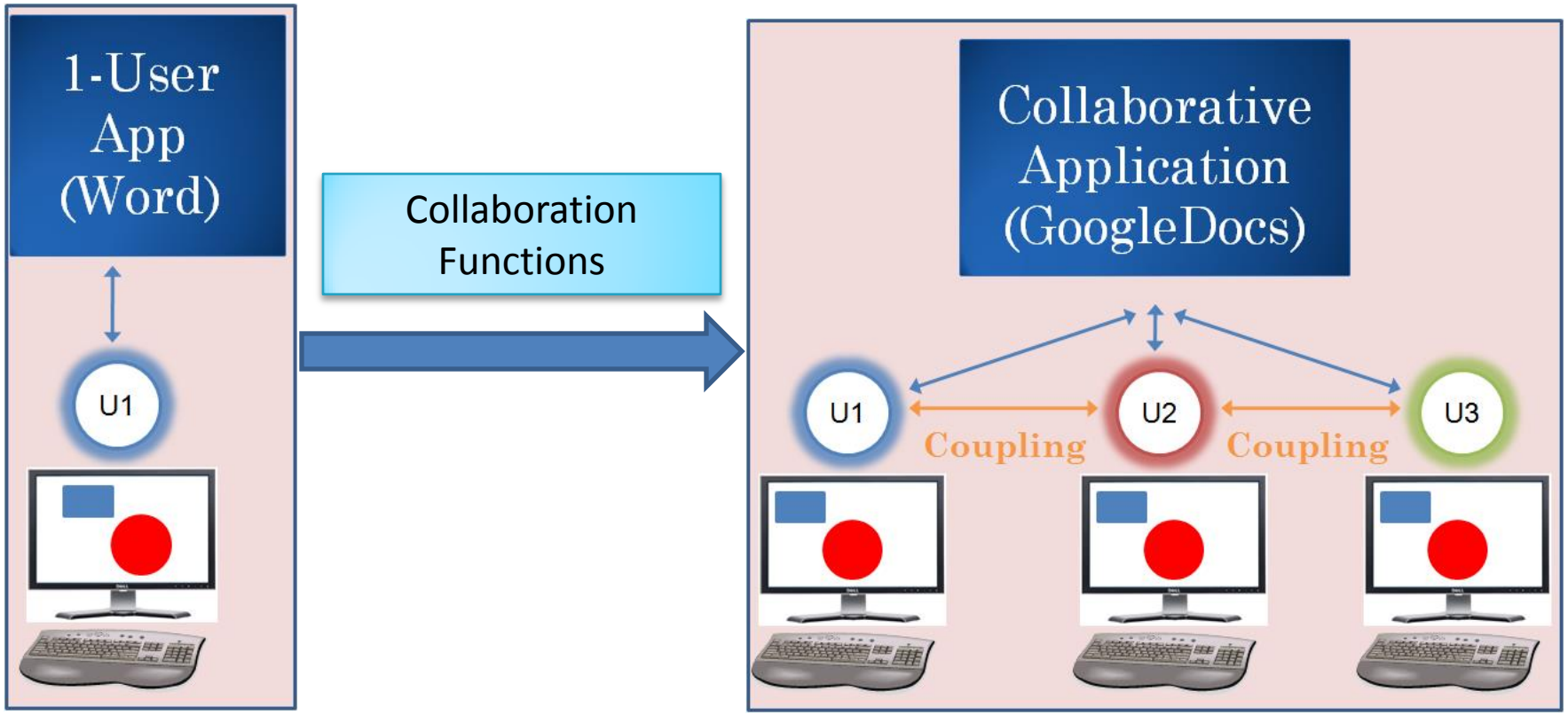
Prasun Dewan
Department of Computer Science
University of North Carolina at Chapel Hill
dewan@cs.unc.edu

Code available at: https://github.com/pdewan/ColabTeaching

Pre-Requisites

- Model-Interactor Separation
- Interaction Types
- Model Types

From 1-user to Collaborative



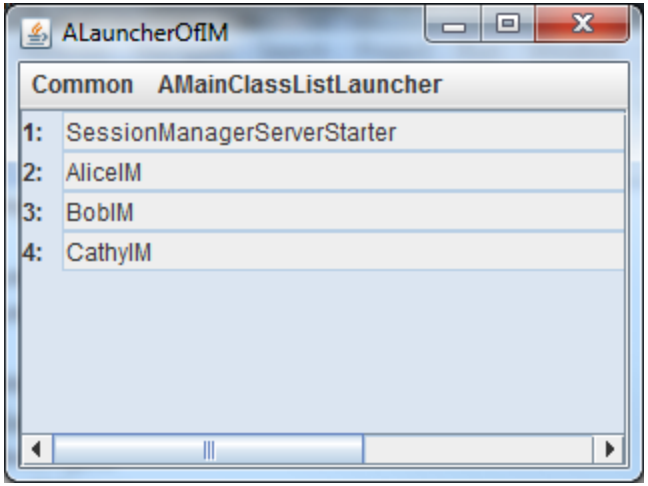
ECHOER TO IM

```
Please enter an input line or quit or history
The woods are lovely dark and deep
The woods are lovely dark and deep
Please enter an input line or quit or history
But I have promises to keep
And miles to go before I sleep
history
The woods are lovely dark and deep, But I have promises to keep, And miles to go before I sleep

Please enter an input line or quit or history
```

```
Please enter an input line or quit or history
The woods are lovely dark and deep
[Alice]The woods are lovely dark and deep
Please enter an input line or quit or history
[Bob]But I have promises to keep
[Cathy]And miles to go before I sleep
history
[Alice]The woods are lovely dark and deep, [Bob]But I have promises to keep, [Cathy]And miles to go before I sleep

Please enter an input line or quit or history
```

Distributed Demoers

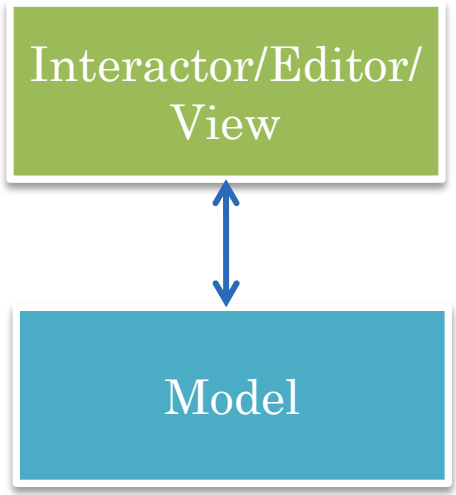
```
public class ALauncherOfIM extends AConsoleModelBasedLauncher
implements LauncherOfIM {
  public Class[] mainClasses() {
    return new Class[] {
       sessionManagerClass(),
       aliceClass(),
       bobClass(),
       cathyClass()
   };
}
```



Can do executeAll or double click on each class in sequence

Can simply run each class from programming environment

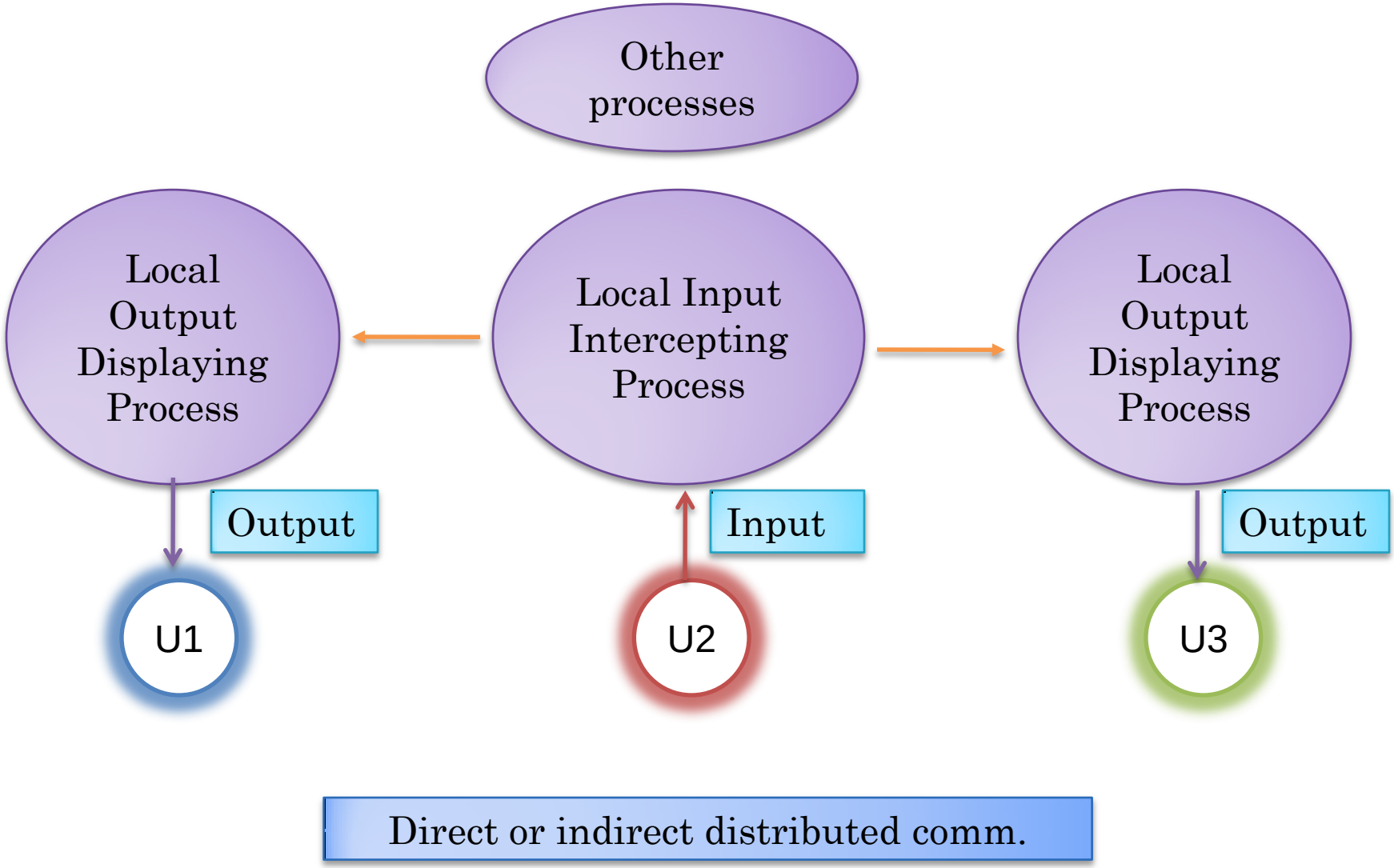
Anatomy /Architecture



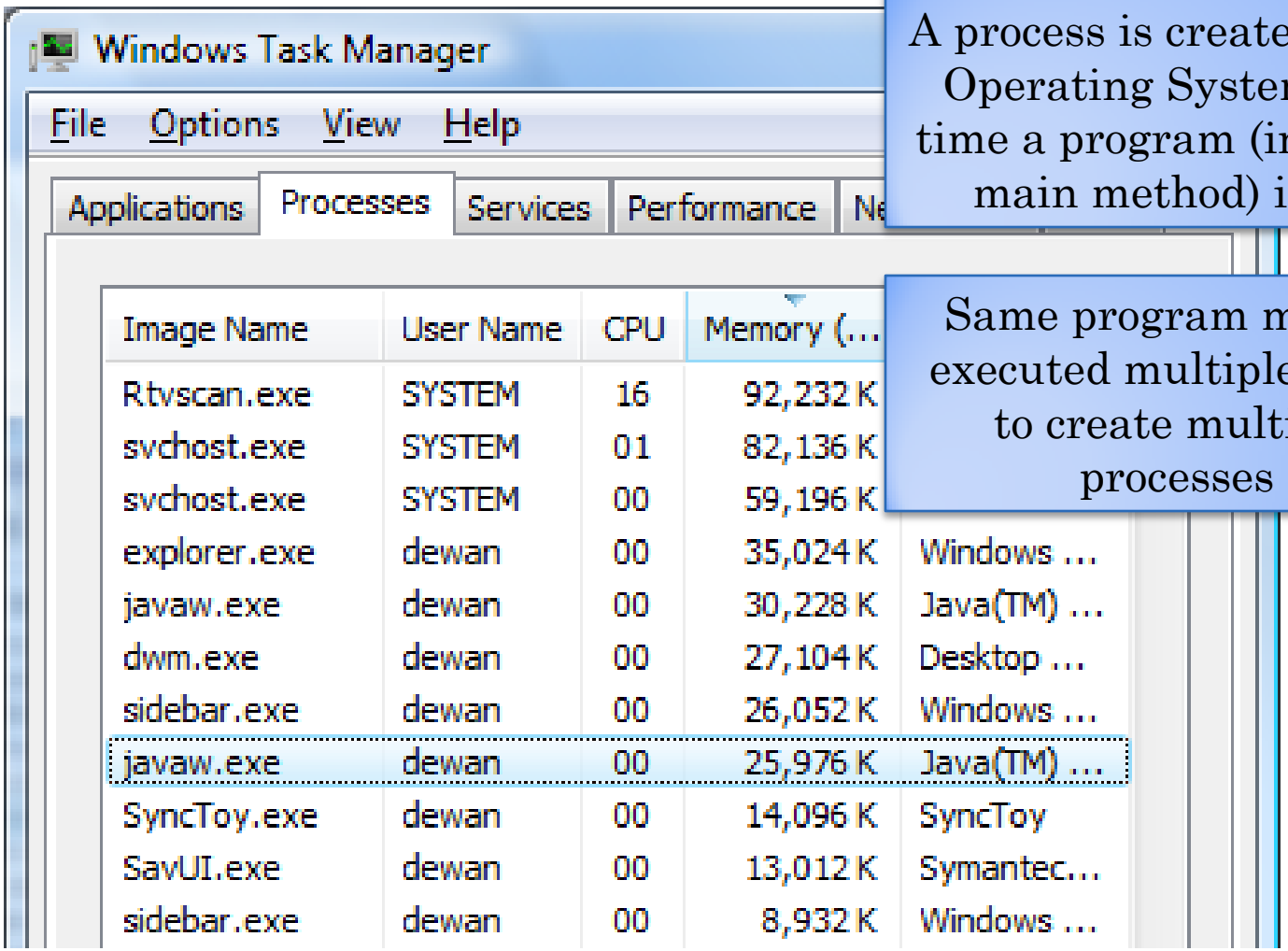
Architecture = Program components and their interaction

Components = objects?

Distributed Processing



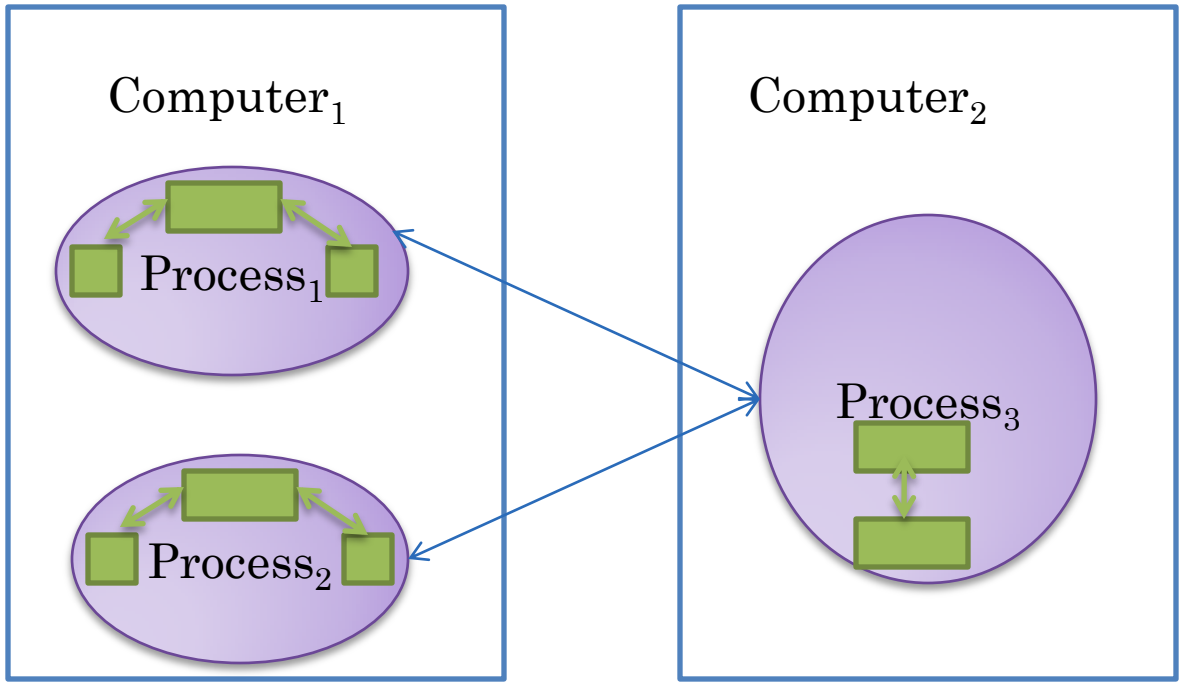
Viewing Processes



A process is created by the Operating System each time a program (in Java a main method) is run

Same program may be executed multiple times to create multiple processes

Process vs Object vs Distributed Architecture



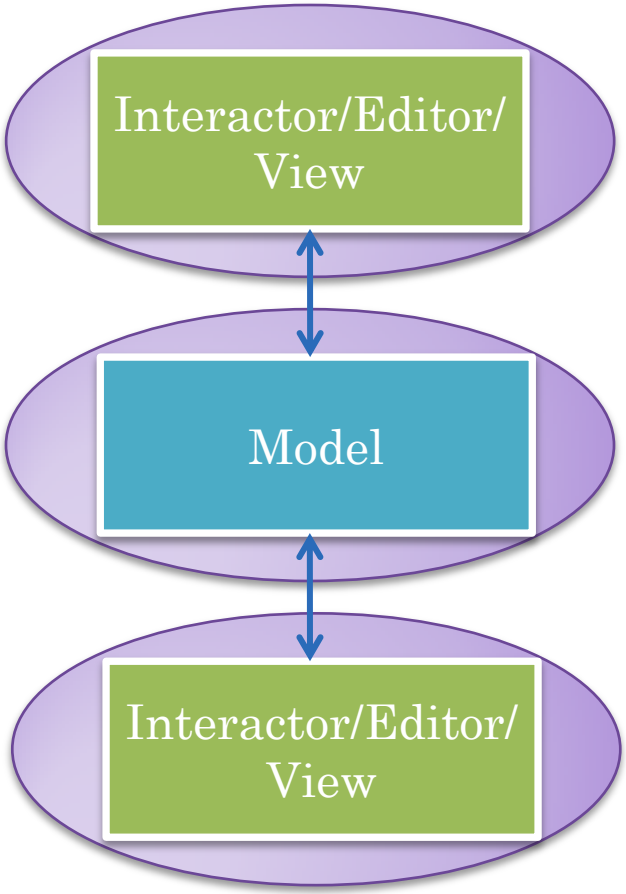
Process architecture describes the processes that implement some potentially distributed application and the communication among these processes

Distributed architecture maps processes to computers

Object architecture describes the objects and the communication among objects in a process

In our demos and testing we will map them all to one machine

Single-User→Collaborative Architecture



Replace local calls with "transparent" remote calls?

Remote calls are not trasparent – must at least deal with communication errors

Blocking call and round trip delay to get local feedback

Central bottleneck which may not always be available

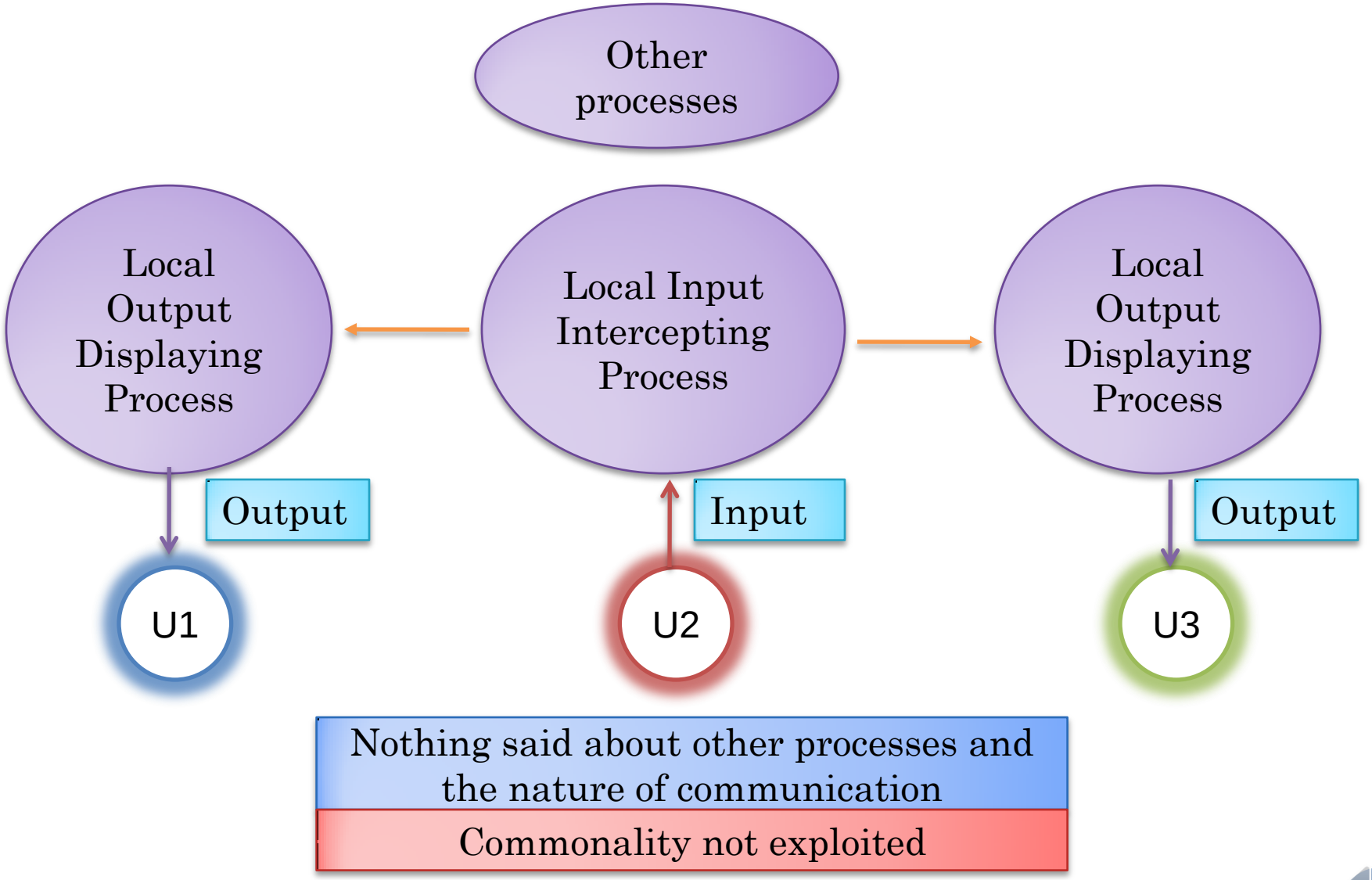
No awareness of others

Sharing at the model level

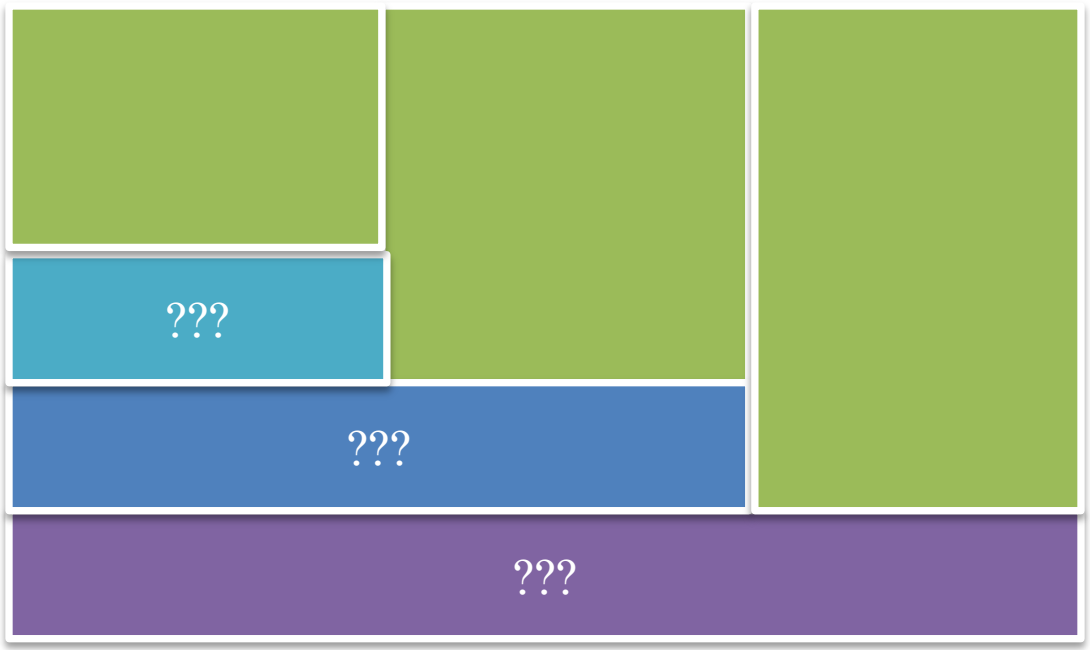
Put the model on one machine and an interactor of a user on his/her machine

Architecture is too constrained!

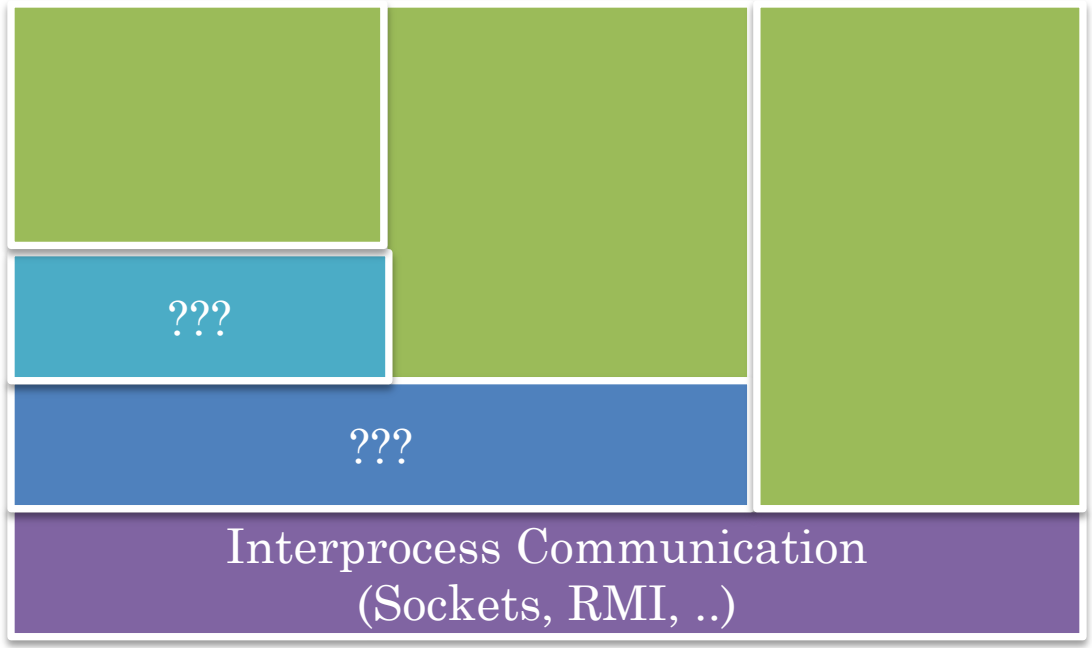
Arbitrary Architecture?



Abstraction Layers in Collaboration Toolkit?



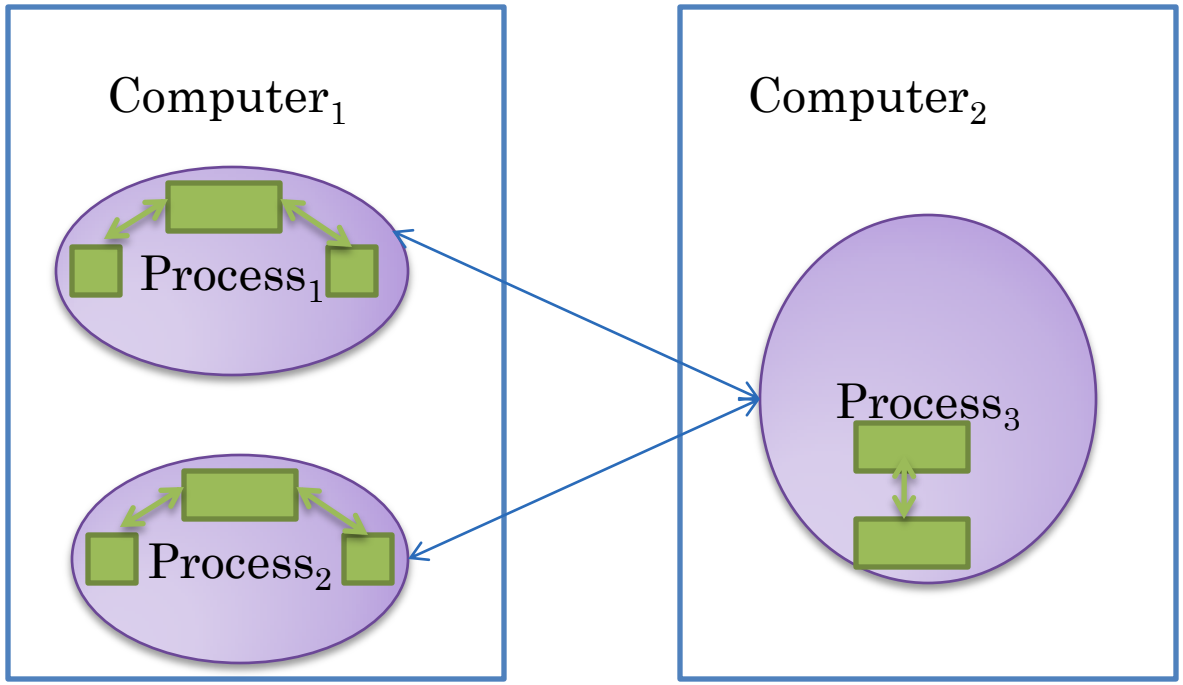
Abstraction Layers in Collaboration Toolkit?



Inter-process communication allows two processes on different hosts to communicate with each other

General form: send (host, port, message)

Process vs Object vs Distributed Architecture (Review)



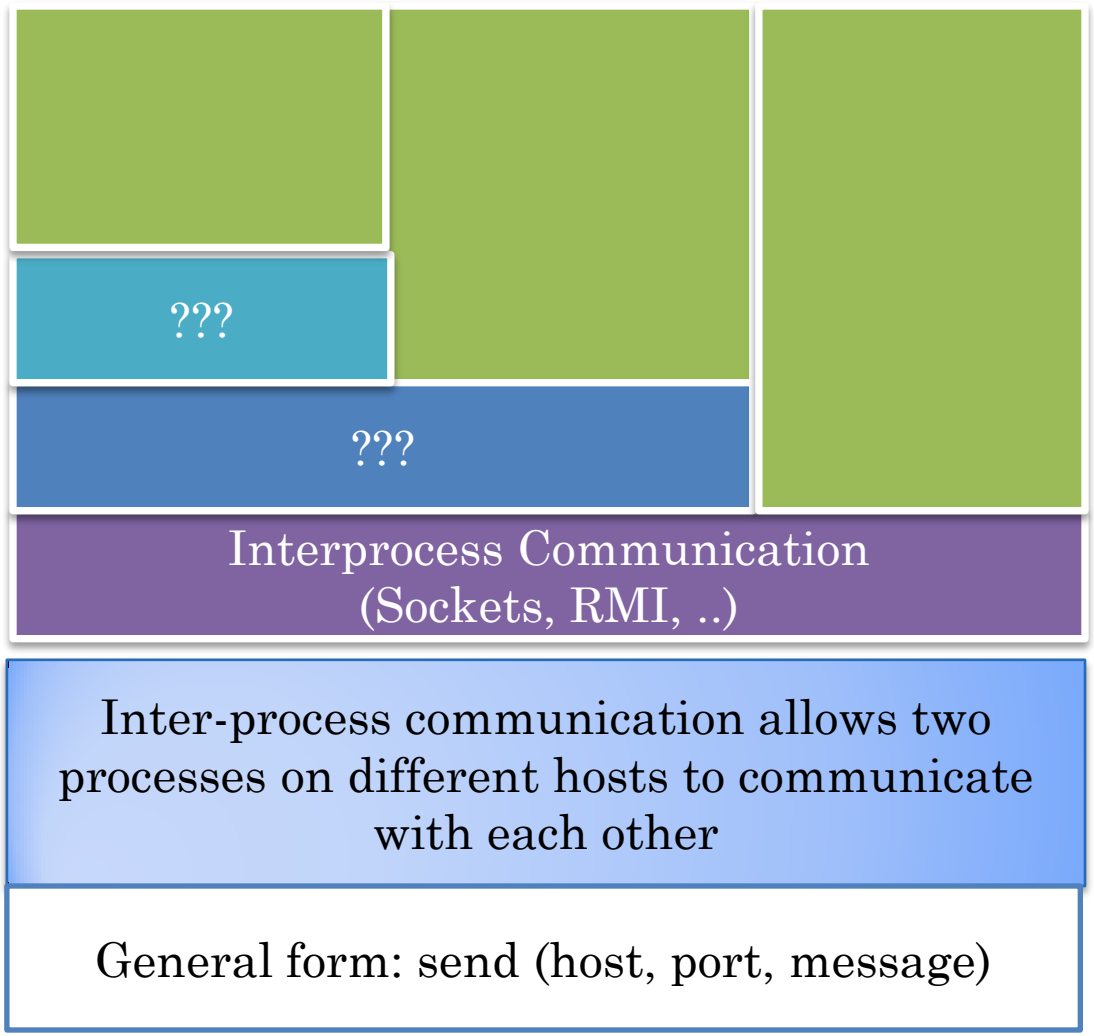
Process architecture describes the processes that implement some potentially distributed application and the communication among these processes

Distributed architecture maps processes to computers

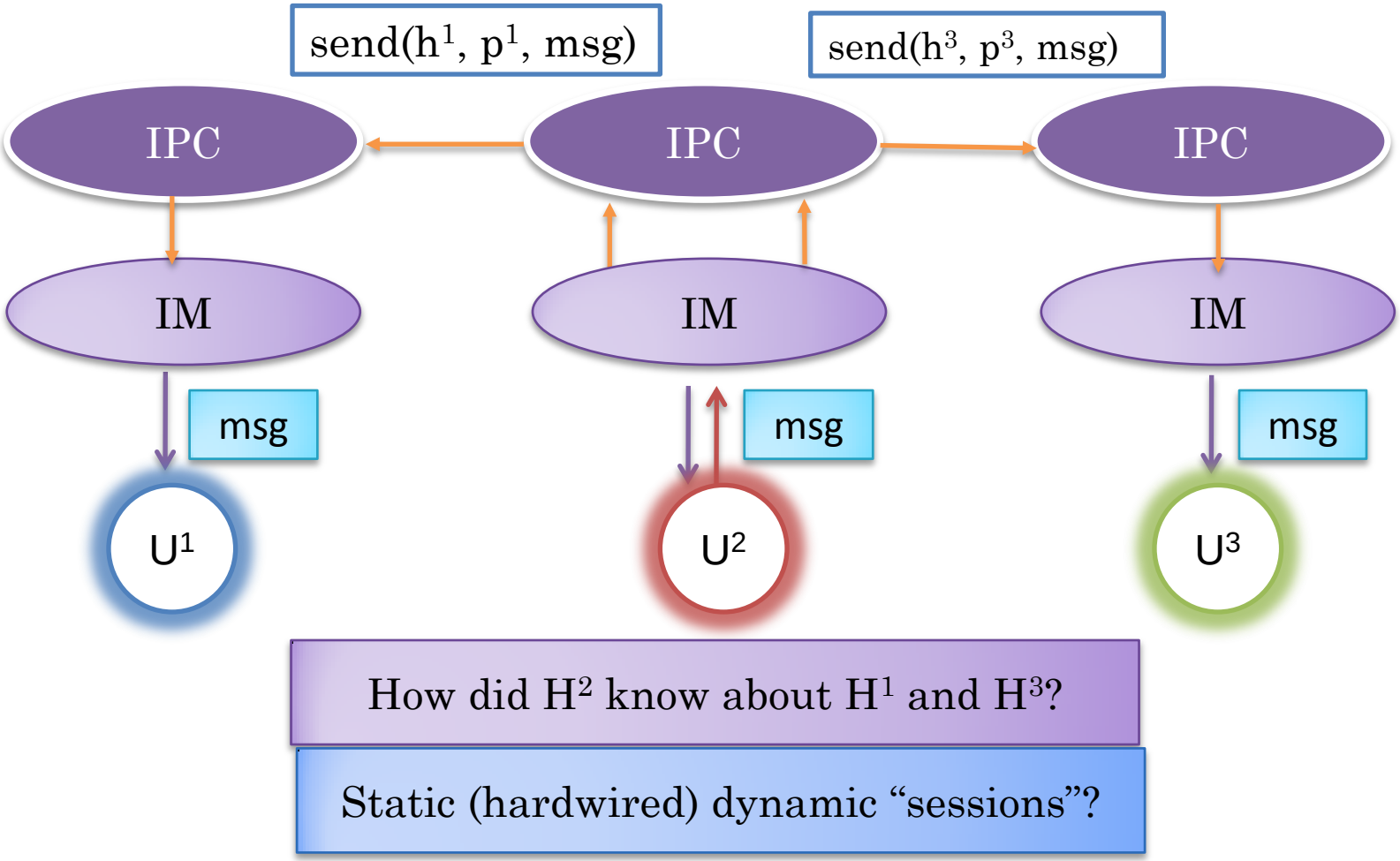
Object architecture describes the objects and the communication among objects in a process

In our demos and testing we will map them all to one machine

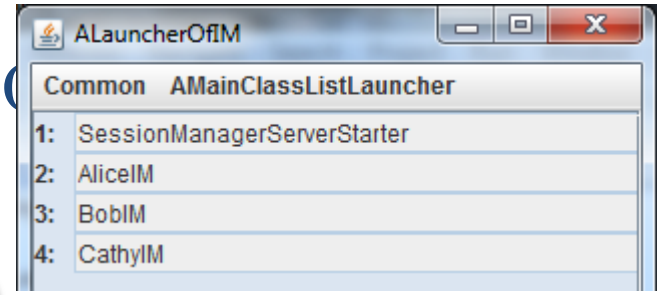
Architecture vs. Distributed Abstractions

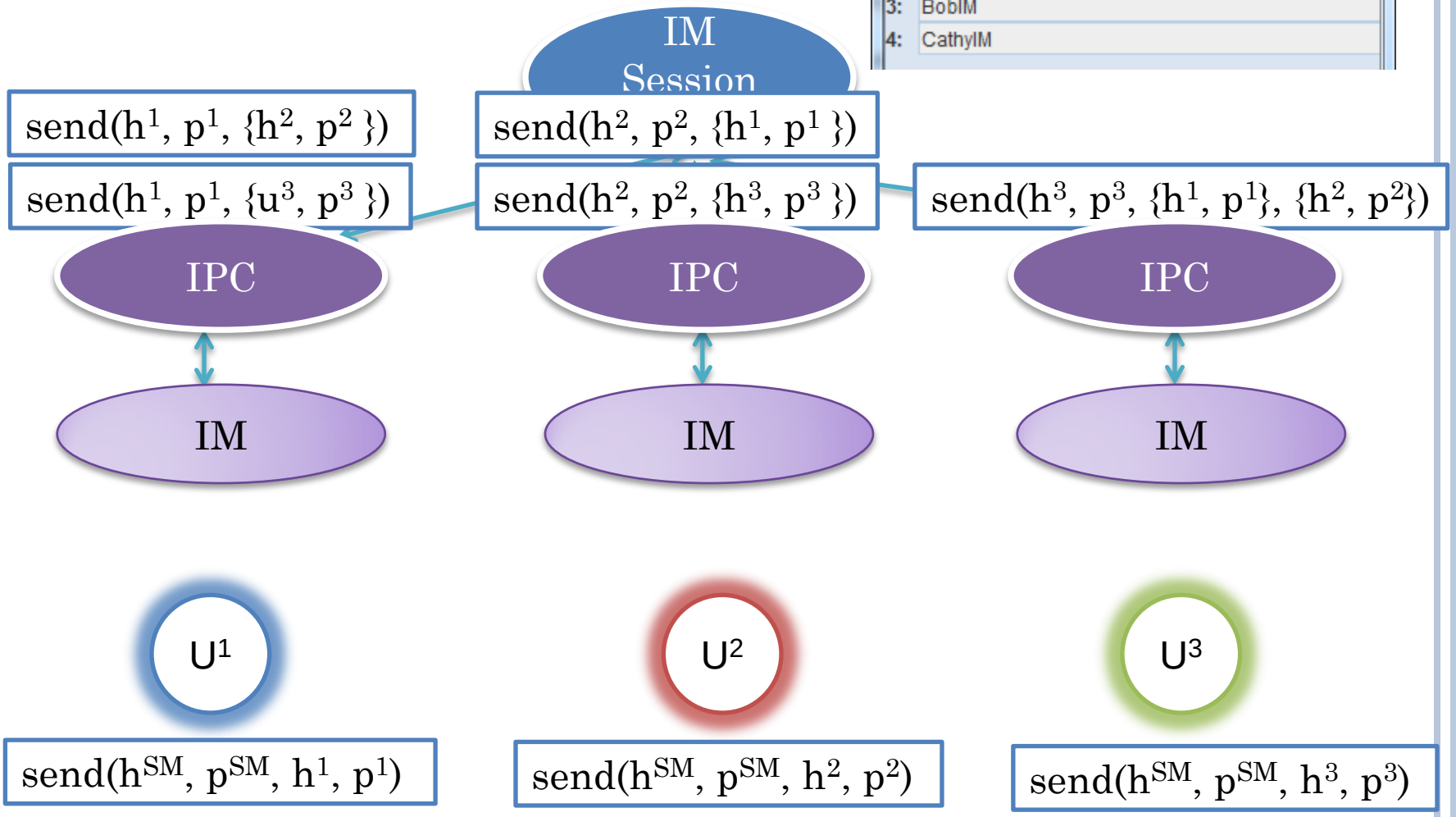


Building IM Using IPC

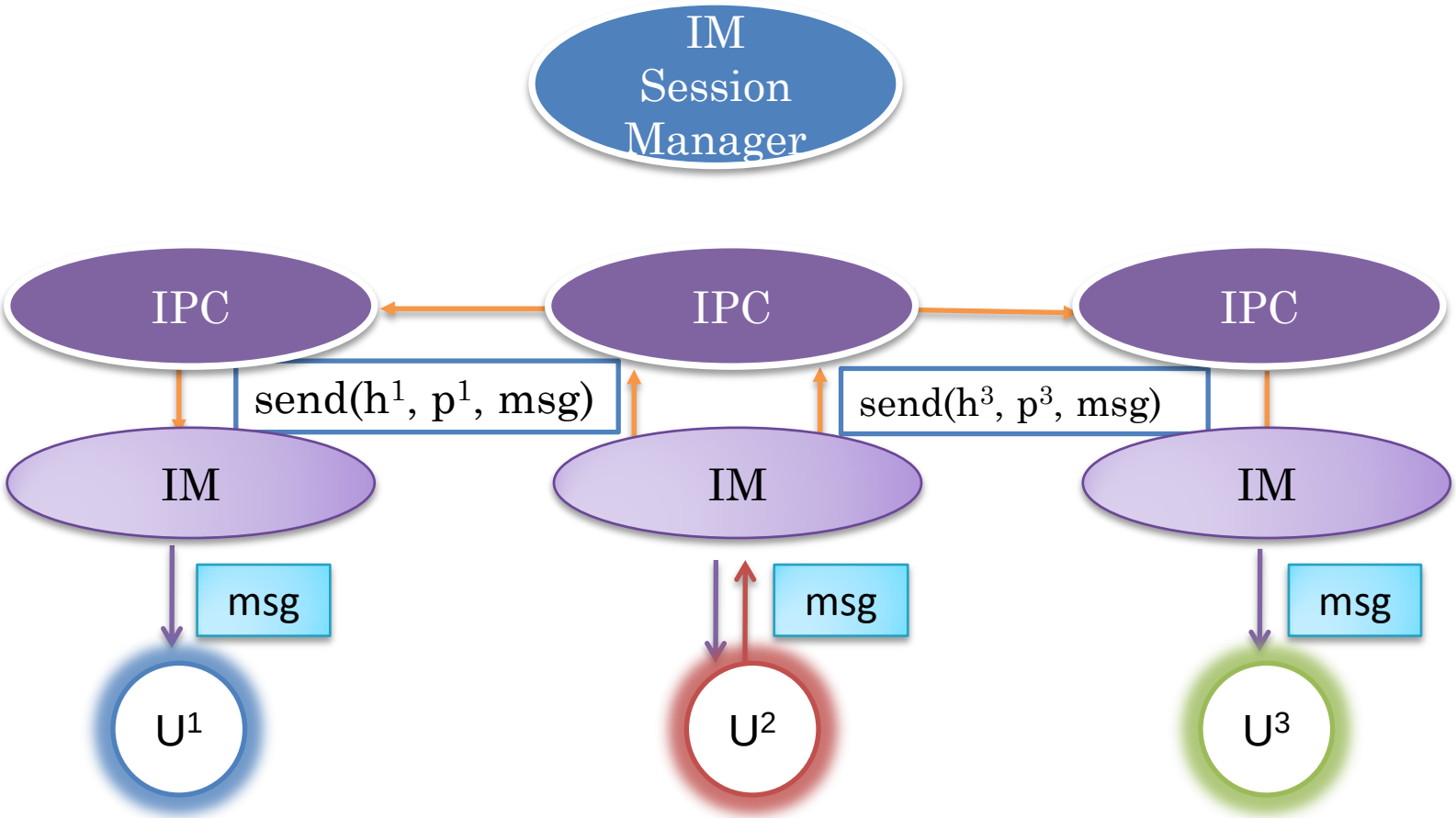


Dynamic sessi

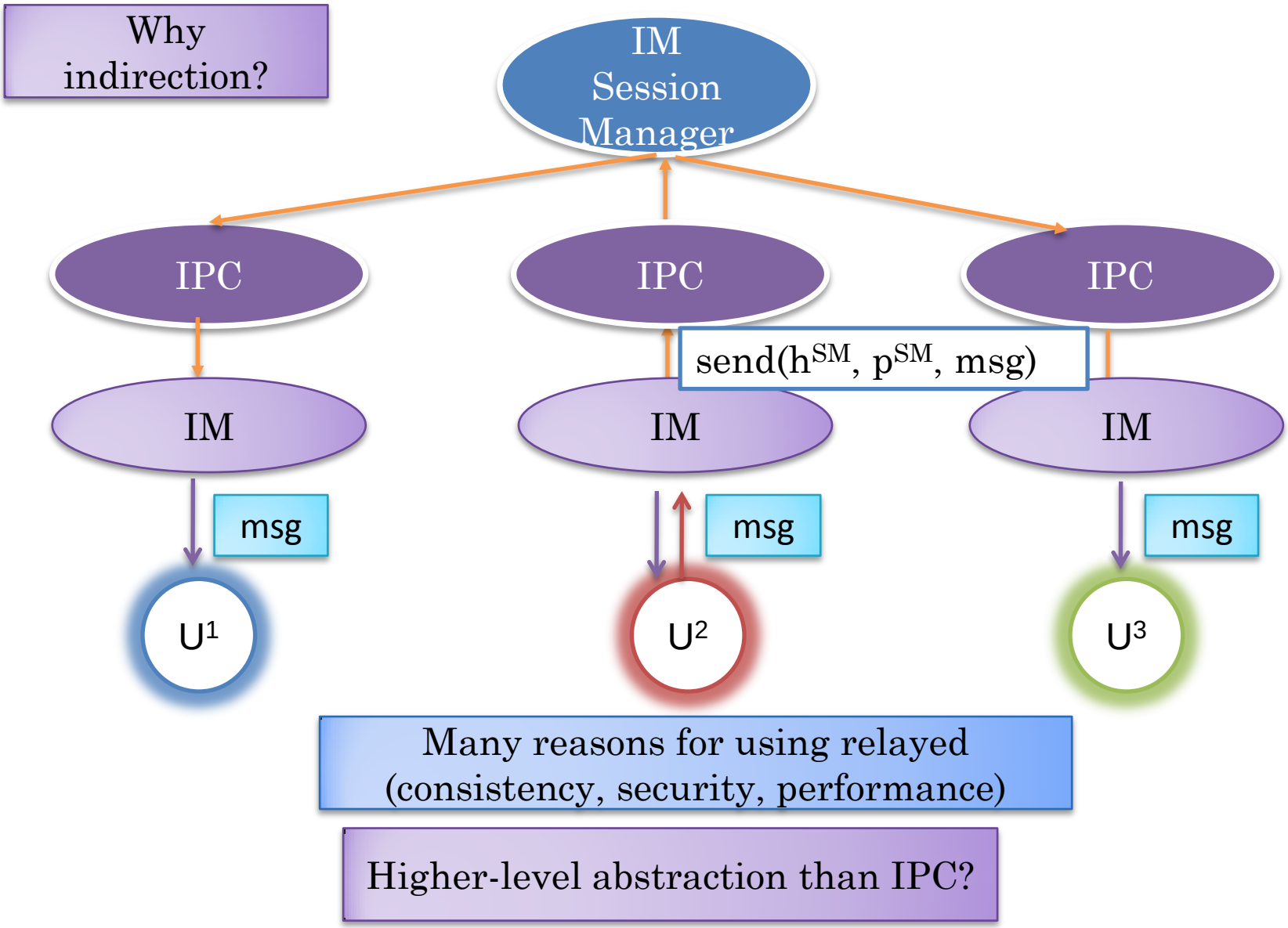




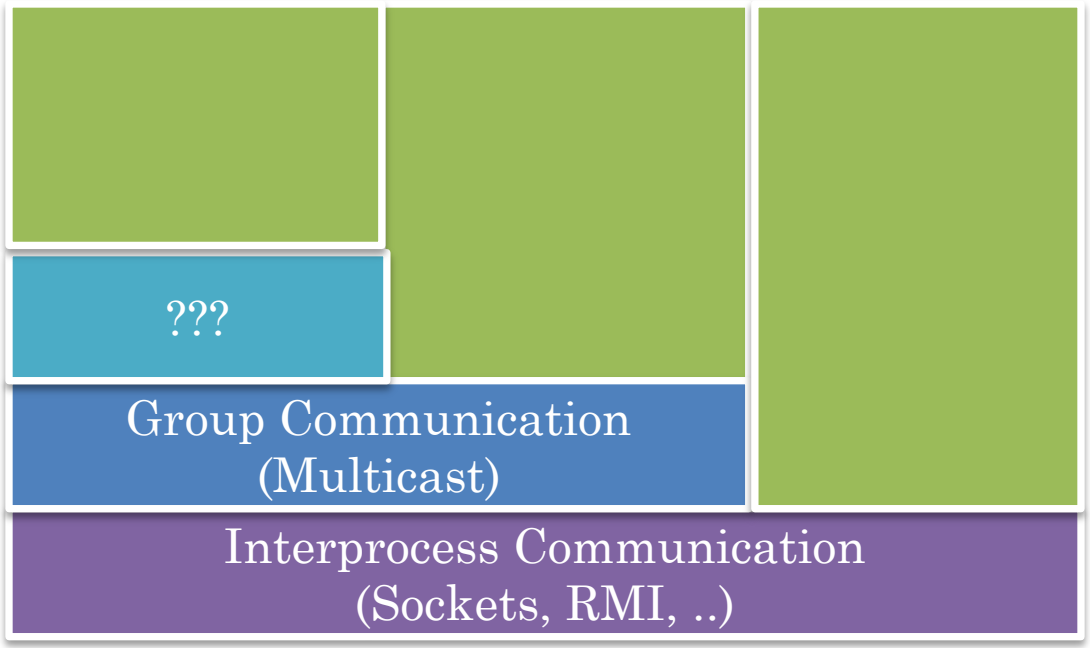
Dynamic P2P


IM
Session
Manager
IPC
IPC
IPC
send(h1, p1, msg)
send(h3, p3, msg)
IM
IM
IM
msg
msg
msg
U1
U2
U3

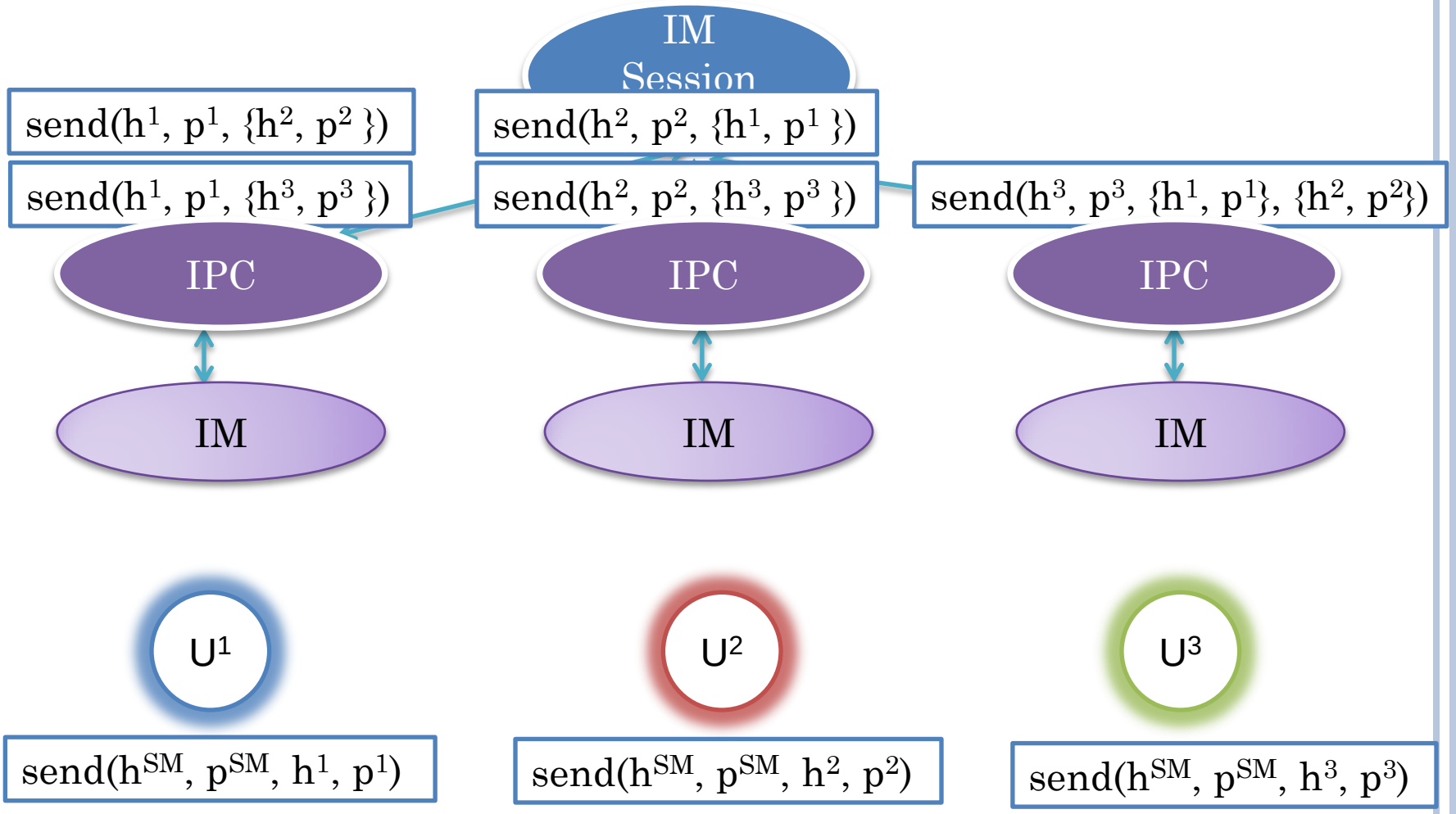
Dynamic Sessions (Relayed)



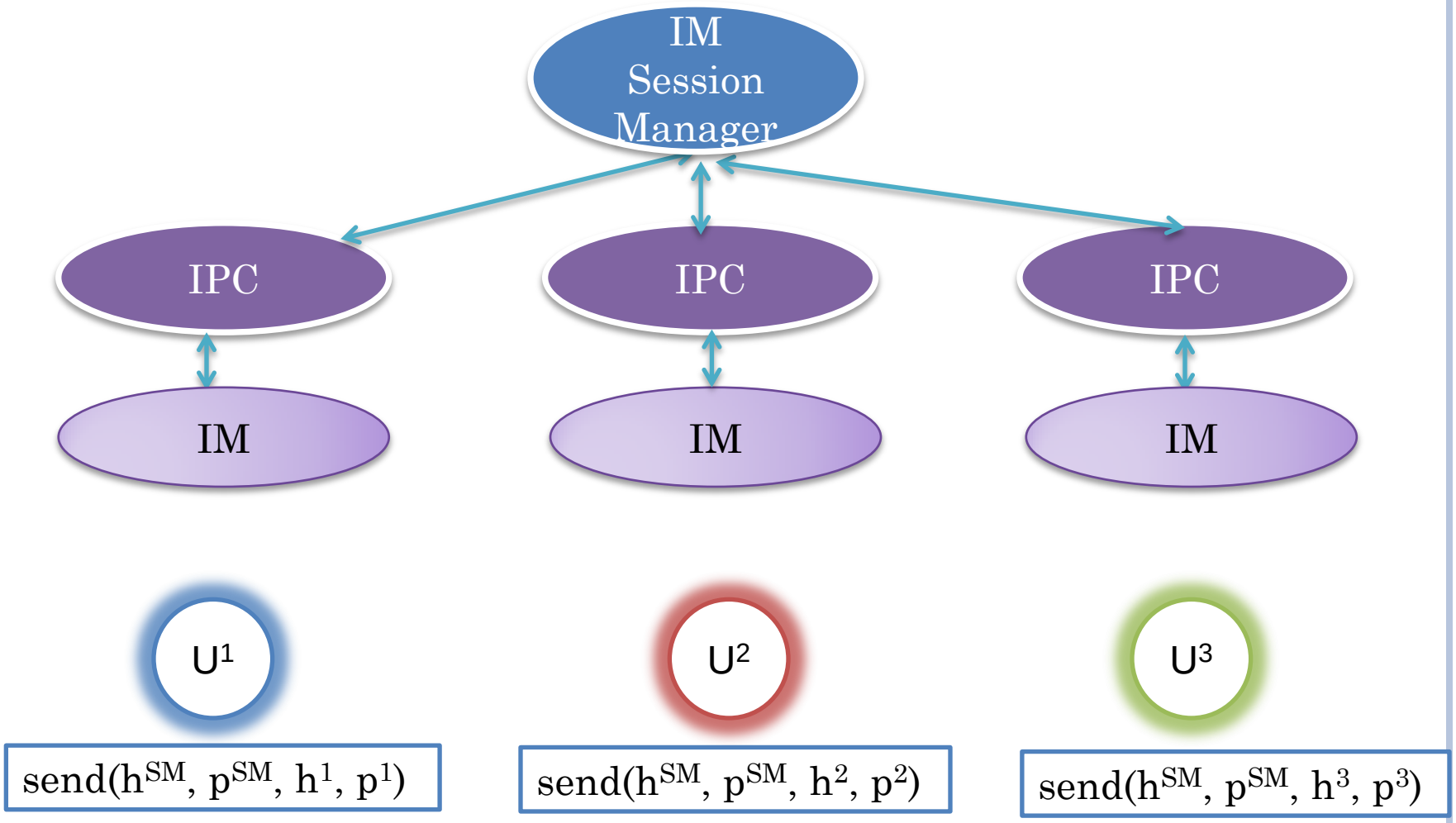
Abstraction Layers in Collaboration Toolkit?



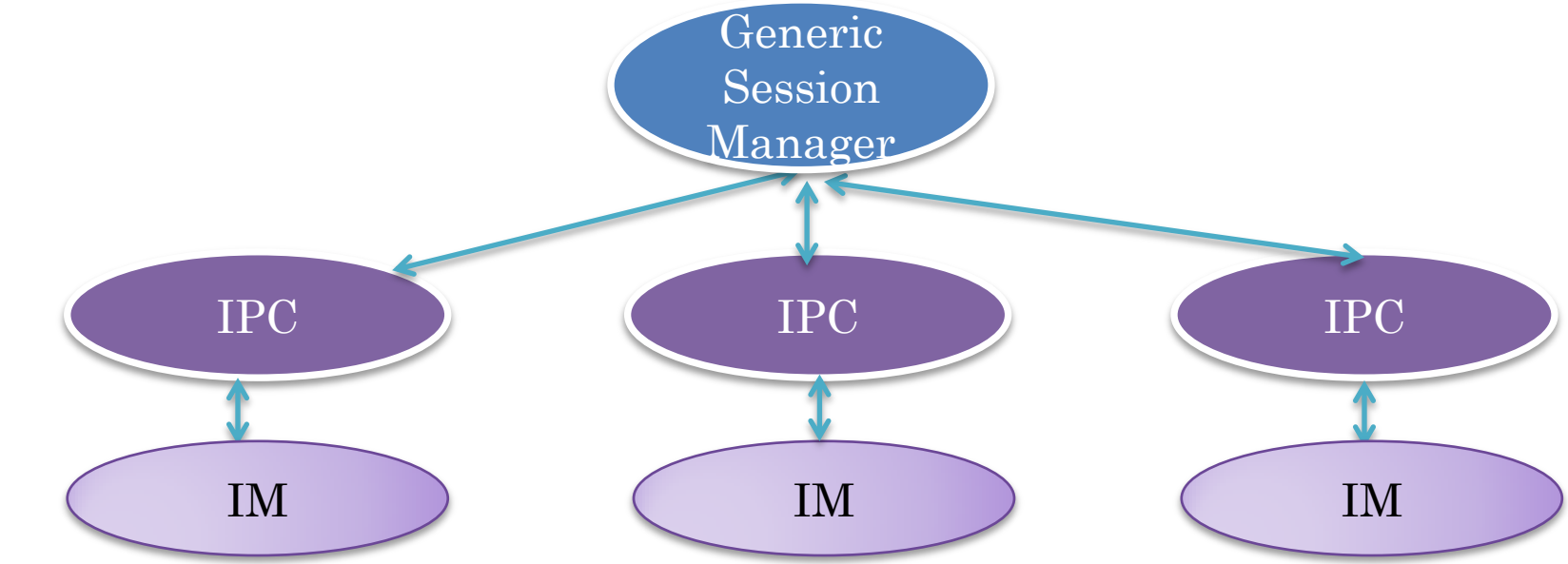
App-Specific Session Manager



App-Specific Session Manager (No Callbacks)



Generic Session Manager (One per Collaborative "Session")



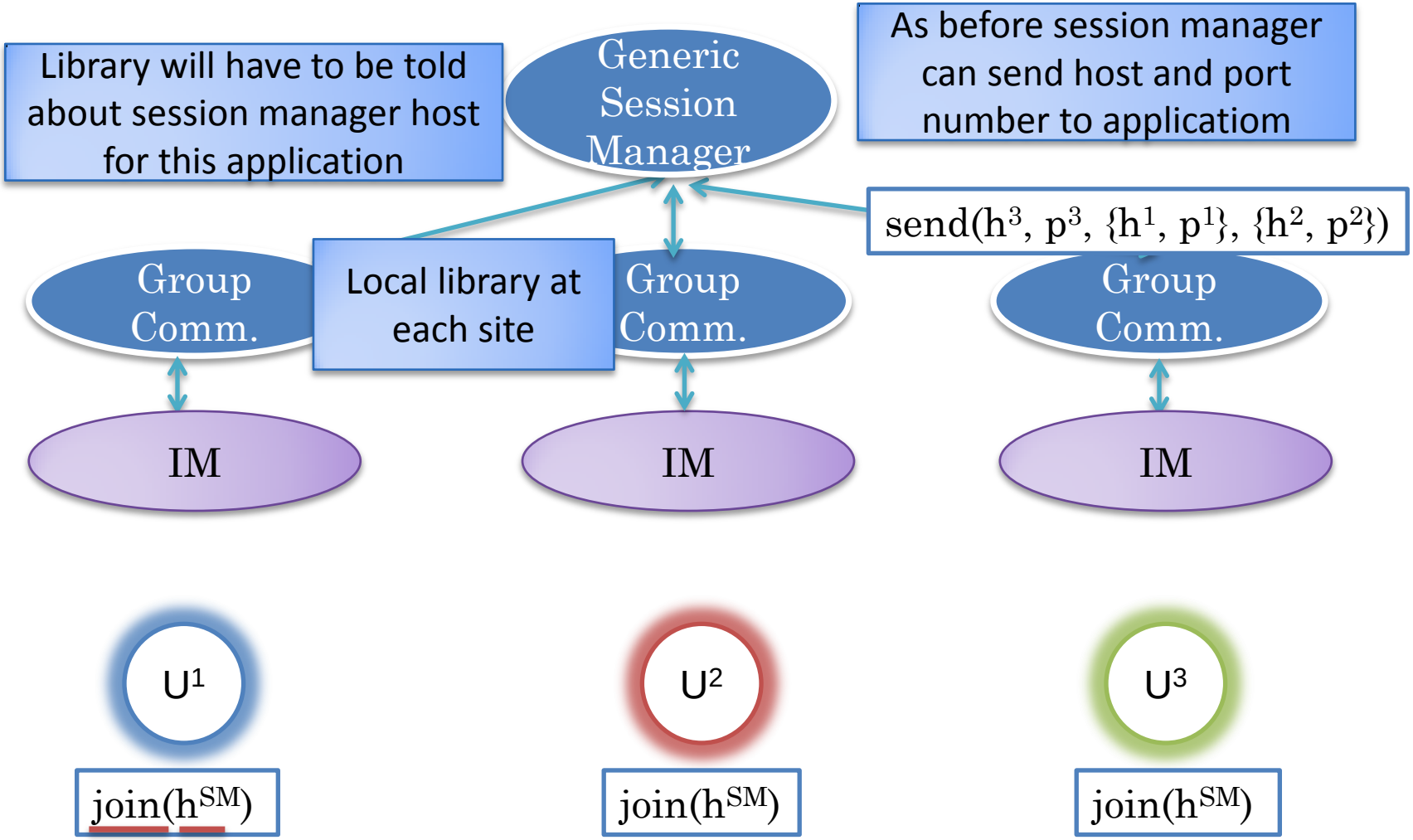
U^1 — send(h^{SM}, p^{SM}, h^1, p^1)

U^2 — send(h^{SM}, p^{SM}, h^2, p^2)

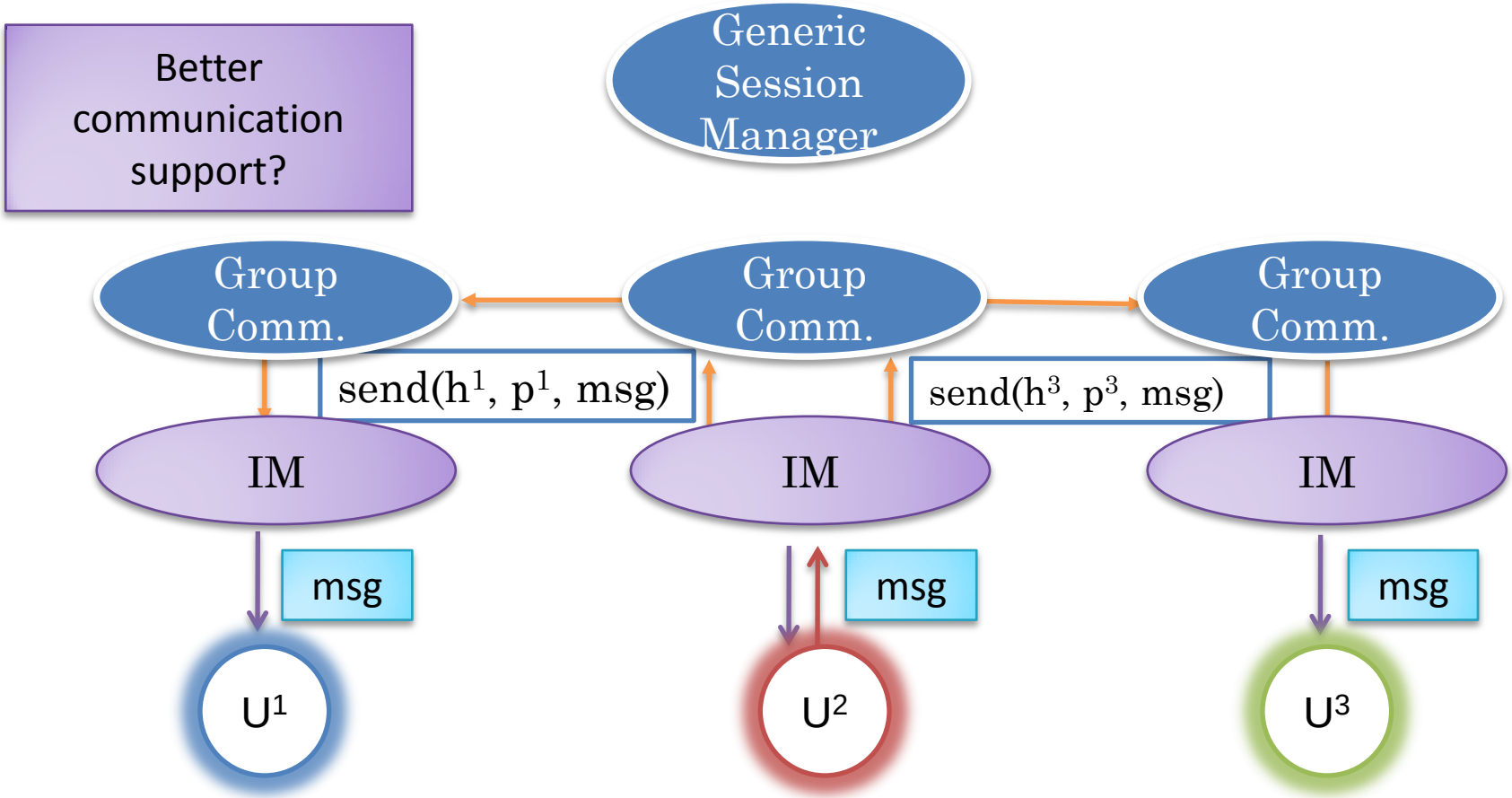
U^3 — send(h^{SM}, p^{SM}, h^3, p^3)

Should clients have to know about ports or send its host?

Generic Session Manager (Client Library, One per Collaborative Session)



Dynamic P2P, No Multicast

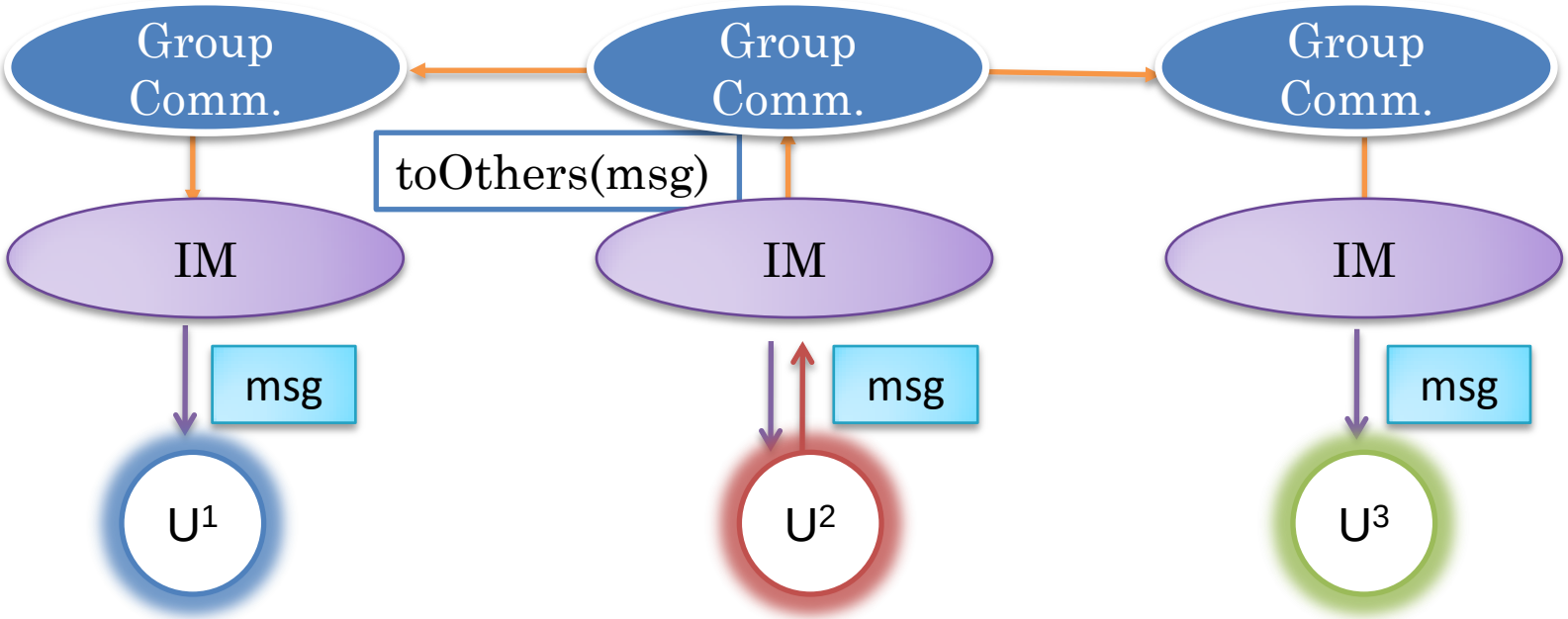


Dynamic P2P Application-Level Multicast

Application-level multicast: Multiple messages delivered to network layer

Generic Session Manager

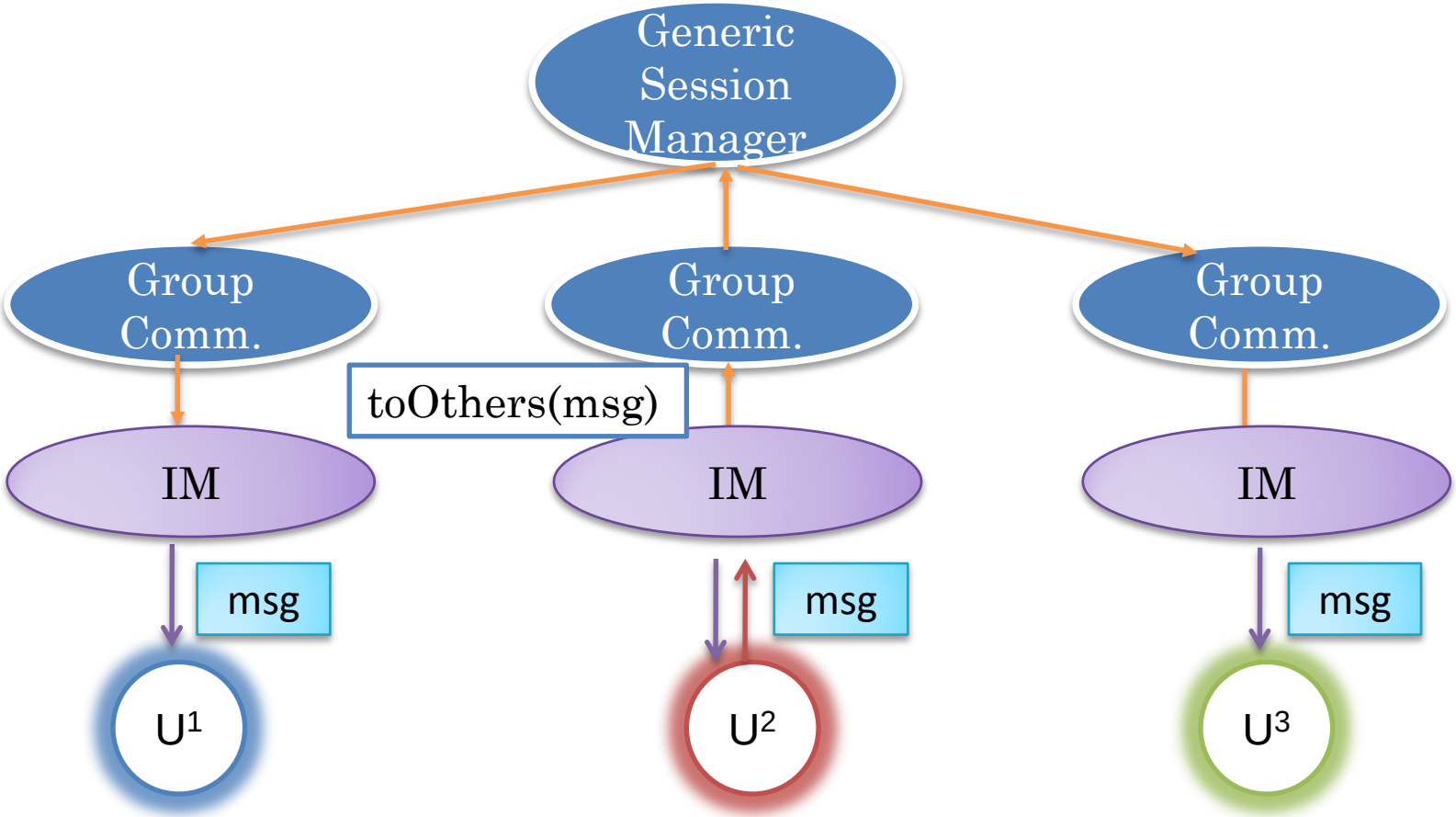
Single message delivered to network layer which can result in a single message being put on the wire for multiple destinations



Application code does not have to define a session participant callback and maintain information about participants

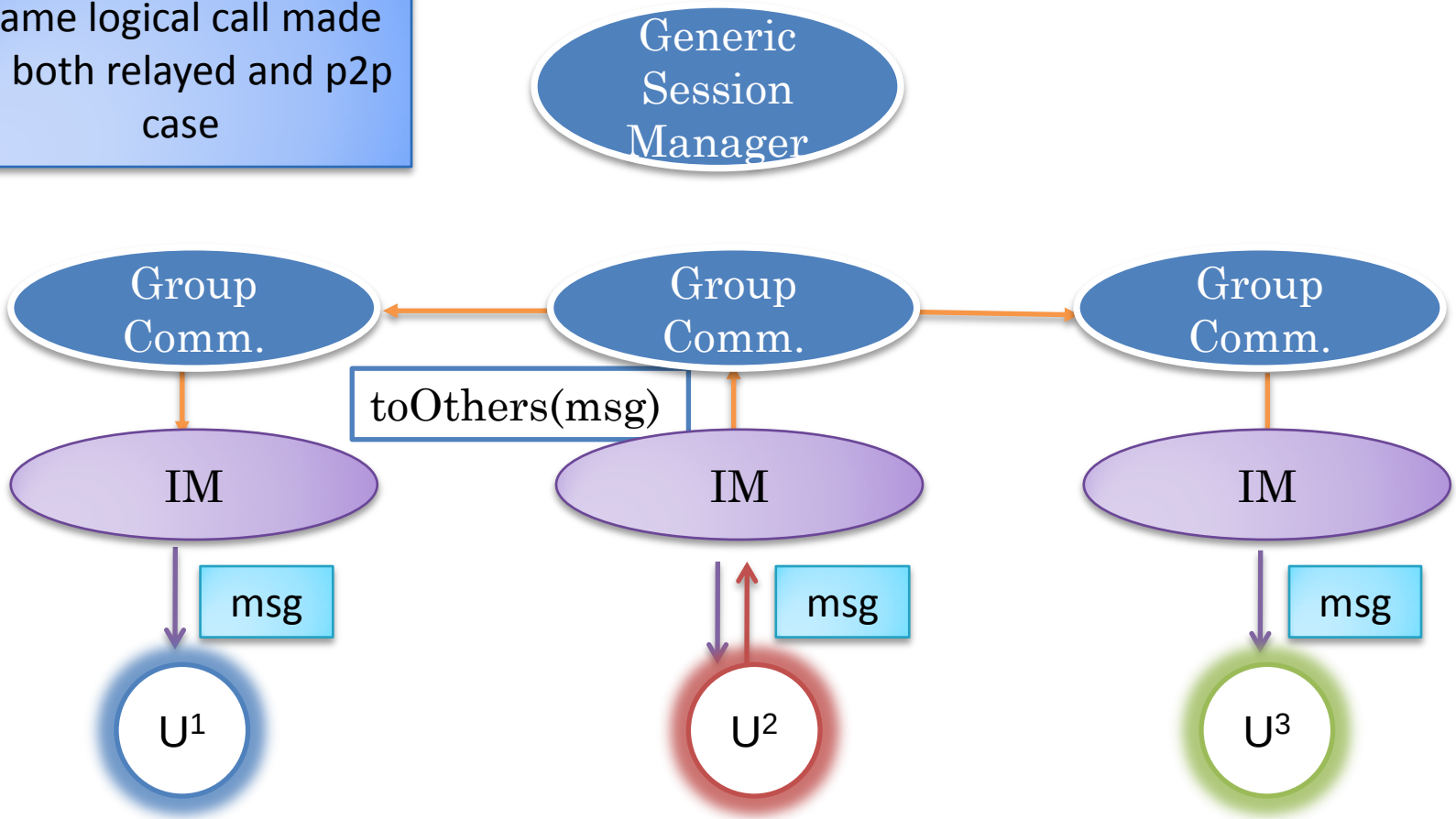
Application code can be unaware of specific users

Dynamic Sessions (Relayed)



Flexible Group Communication

Same logical call made in both relayed and p2p case

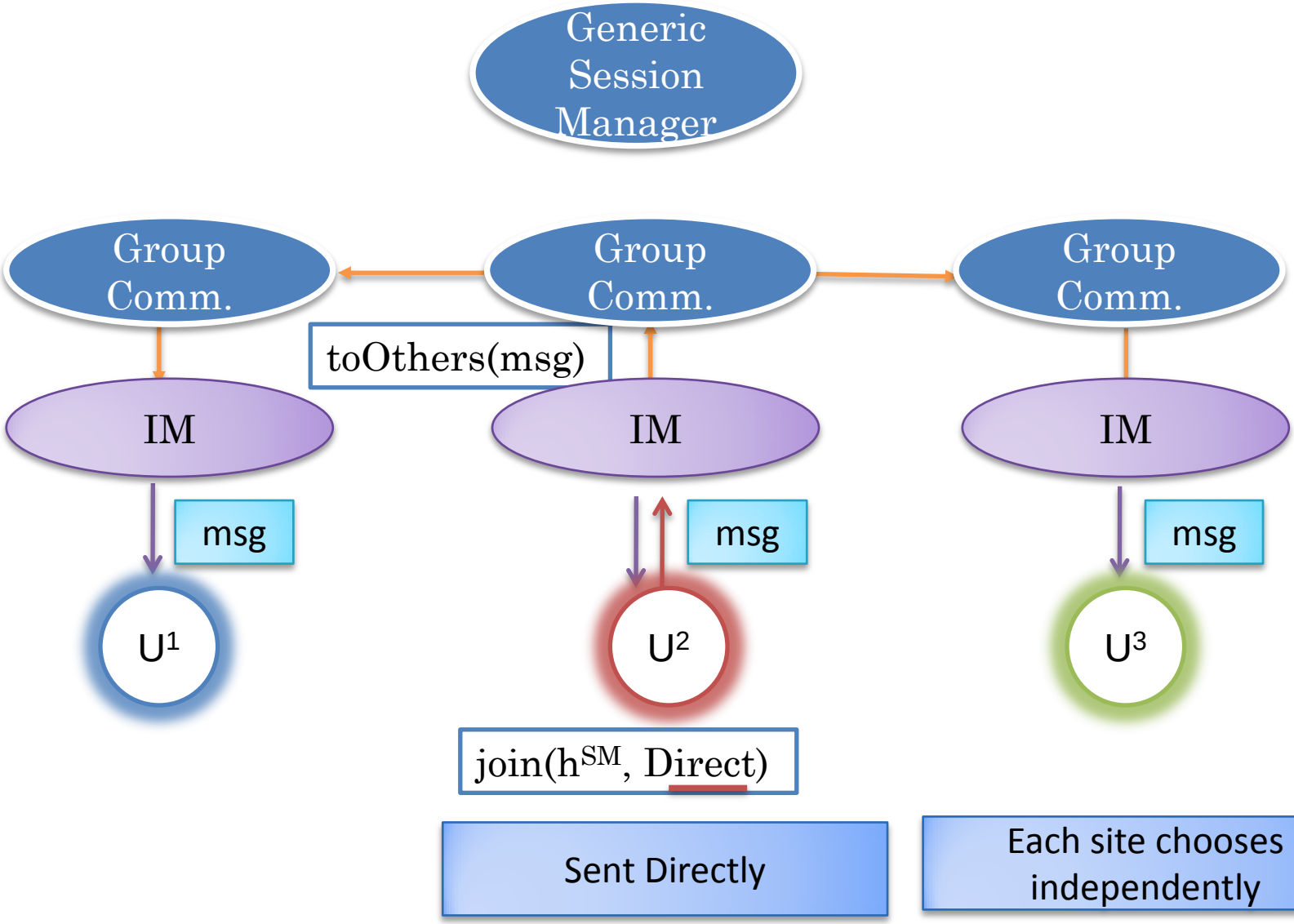


Library can support both forms of communication

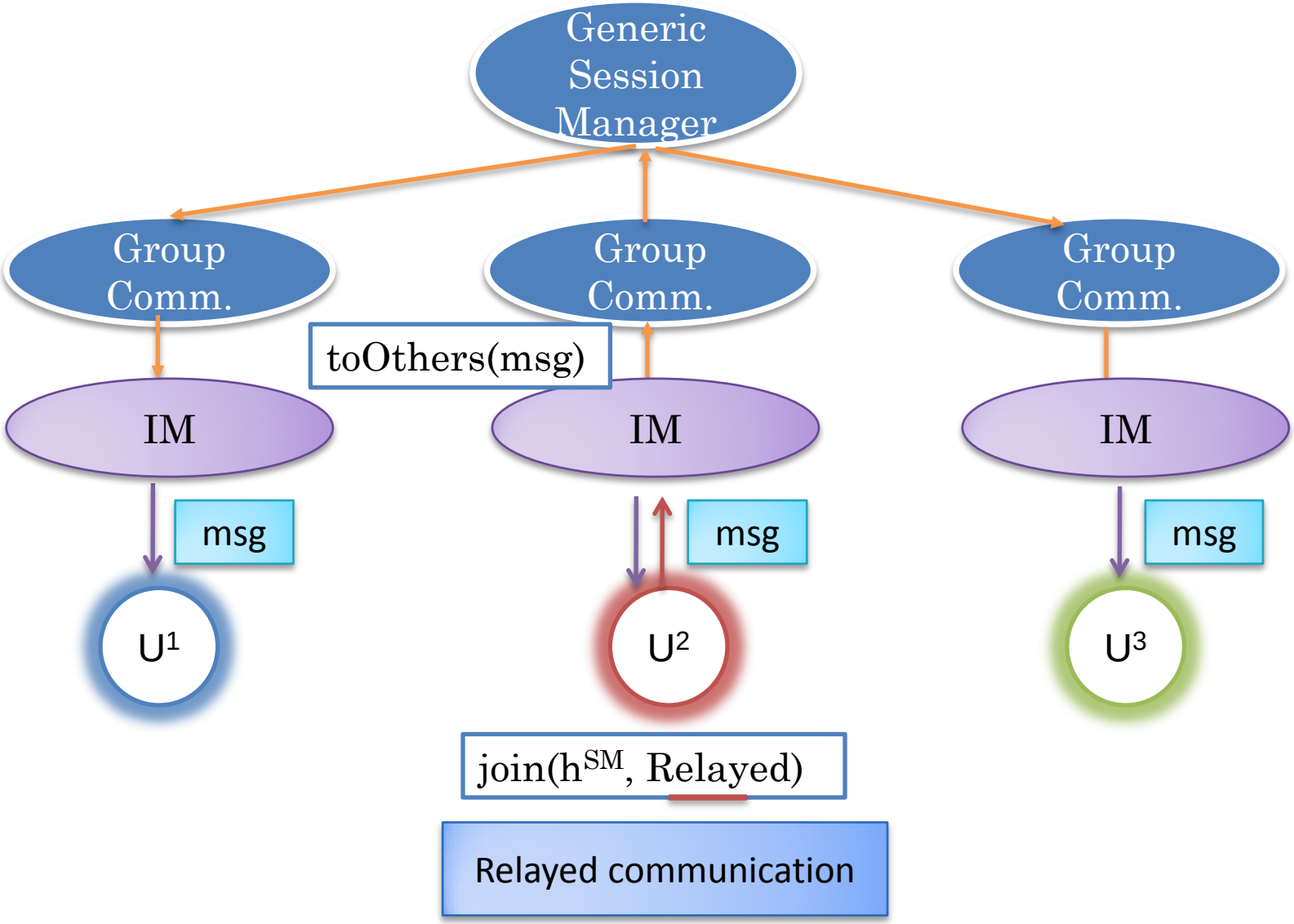
It can choose based on security and performance characteristics (but not consistency)

It can let application choose

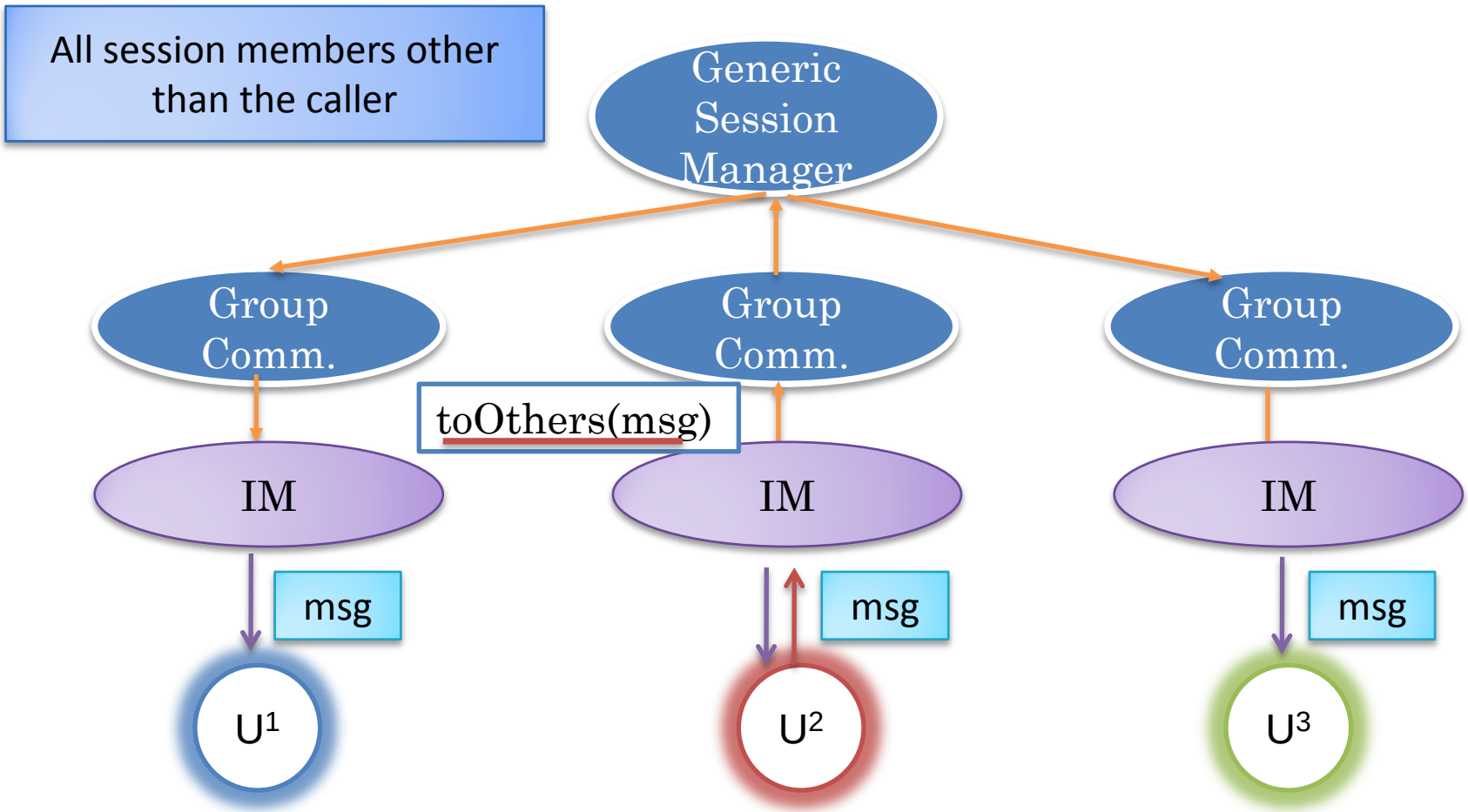
Specifying Route at Join Time: P2P



Specifying Route at Join Time: Relayed

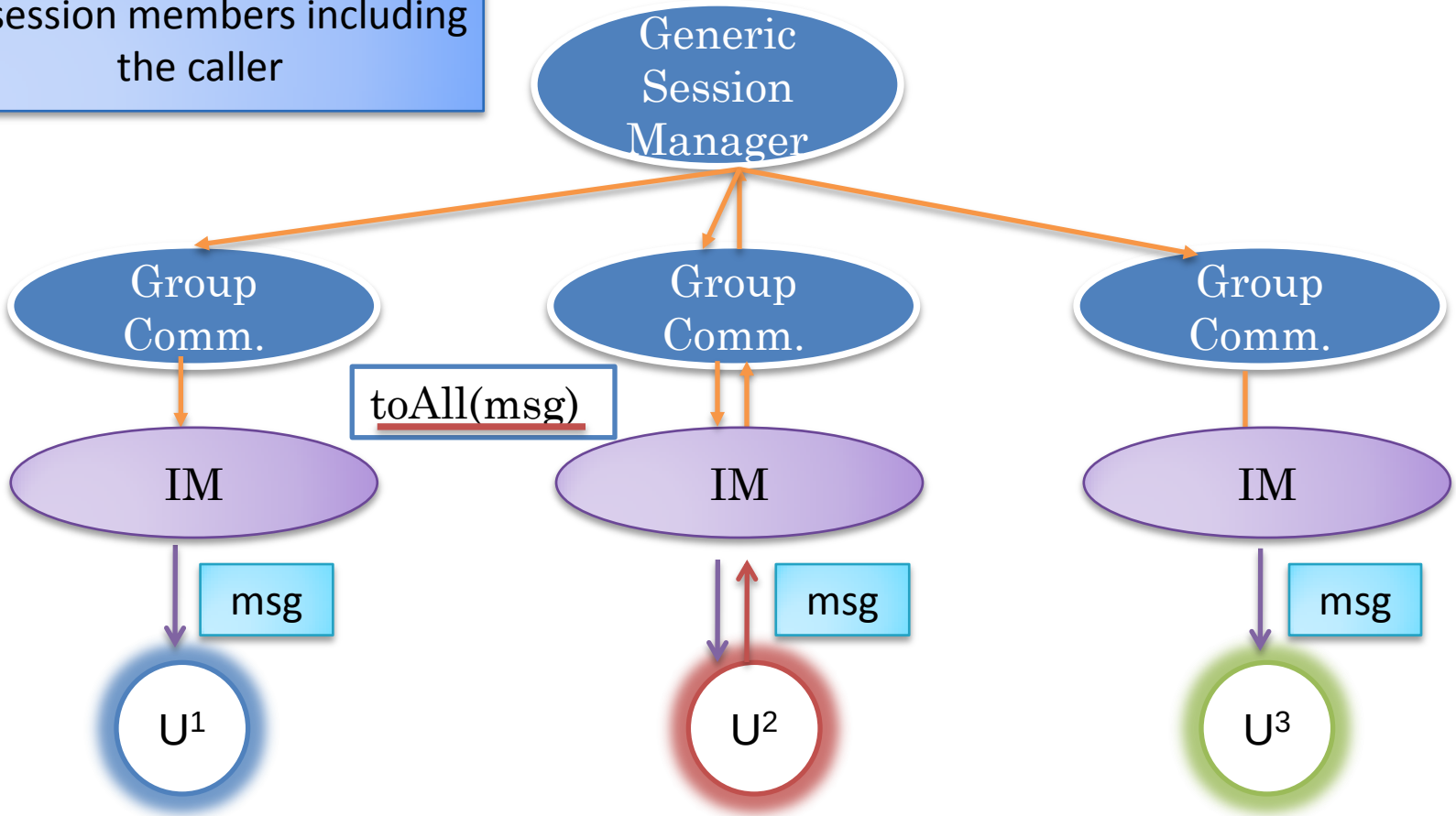


Additional Multicast Groups?



toAll

All session members including the caller



MUD (Multi-User Dungeons): Other Groups

- **say** | " *your utterance here*
  Everyone in the room can 'hear' what you 'say', or see what you type.
- **w**hisper *playername* = *your whisper here*
  ... so only the player(s) named, and in the room, can hear your whisper.
- **mut**ter *player* = *message*
  Mutters *message* to everyone in the same room EXCEPT *player*.

toOthers(msg)

toClient(user, msg)

toClients({u^1, .., u^n}, msg)

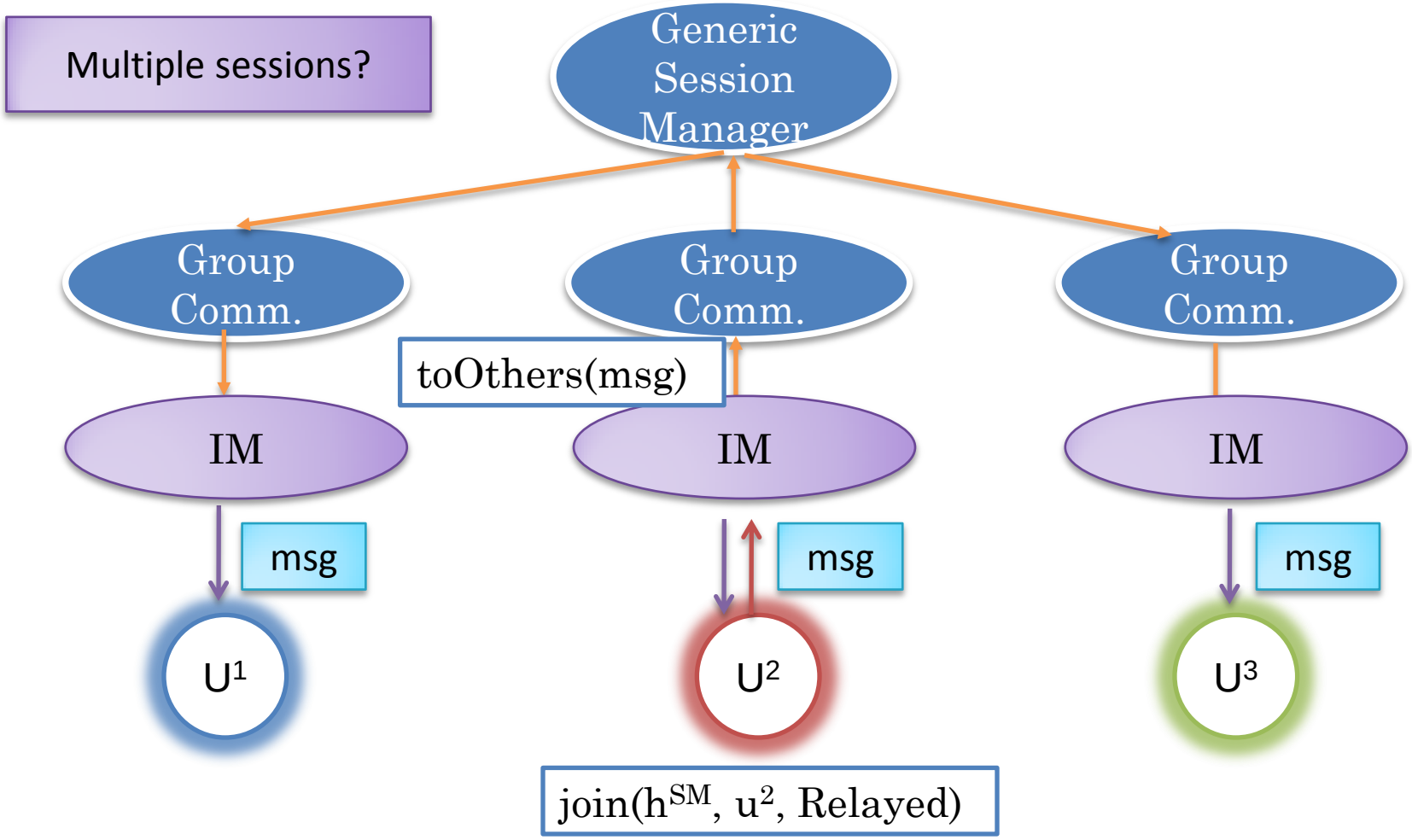
join(h^{SM}, $u^{2,}$ Relayed)

Logical user id (credentials) rather than physical host given as argument

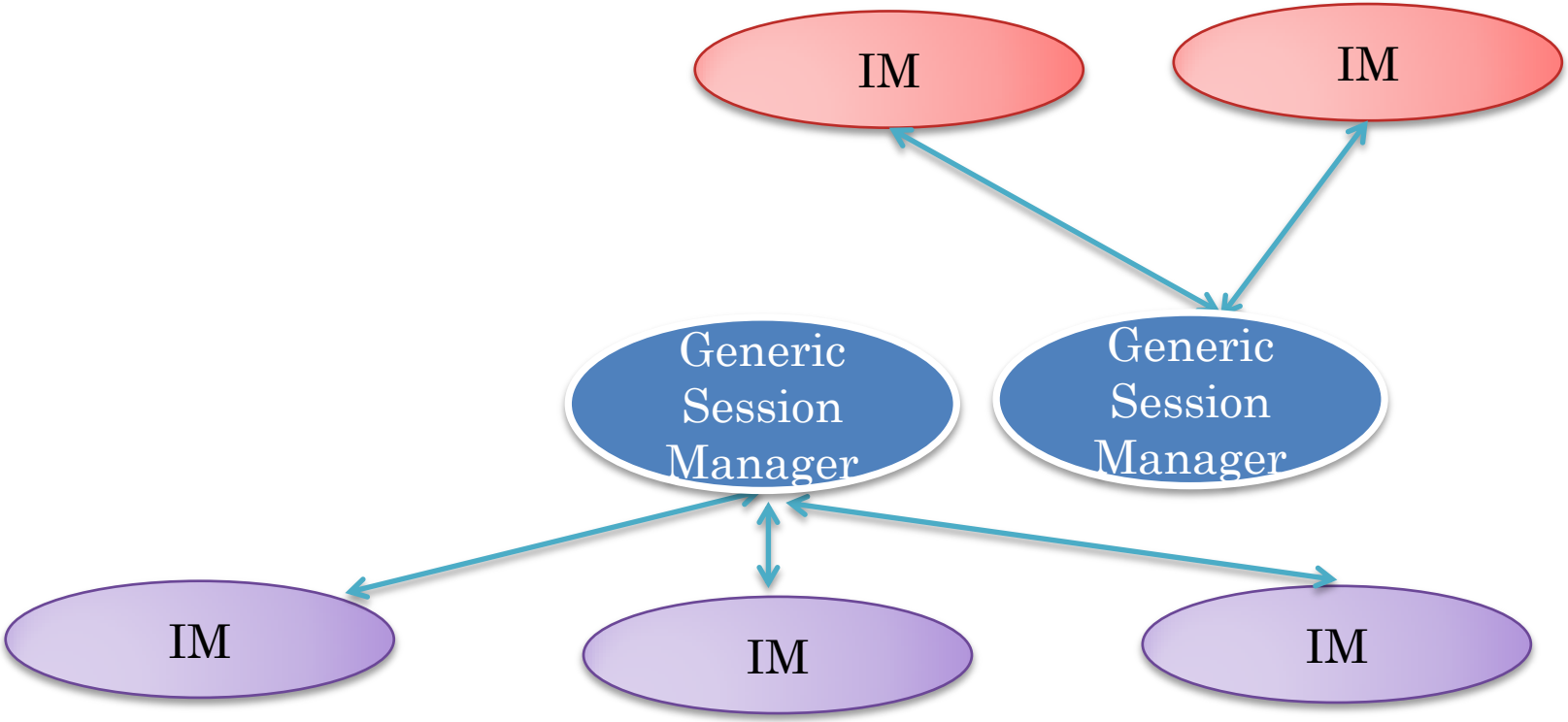
User-aware application code but not host and port aware

Curtis, P. (1992), *Muddings: Social Phenomena in Text-Based Virtual Reality*. 1992: Xerox Palo Alto Research Center.

Single Session per Session Manager



Single Session Per Session Manager

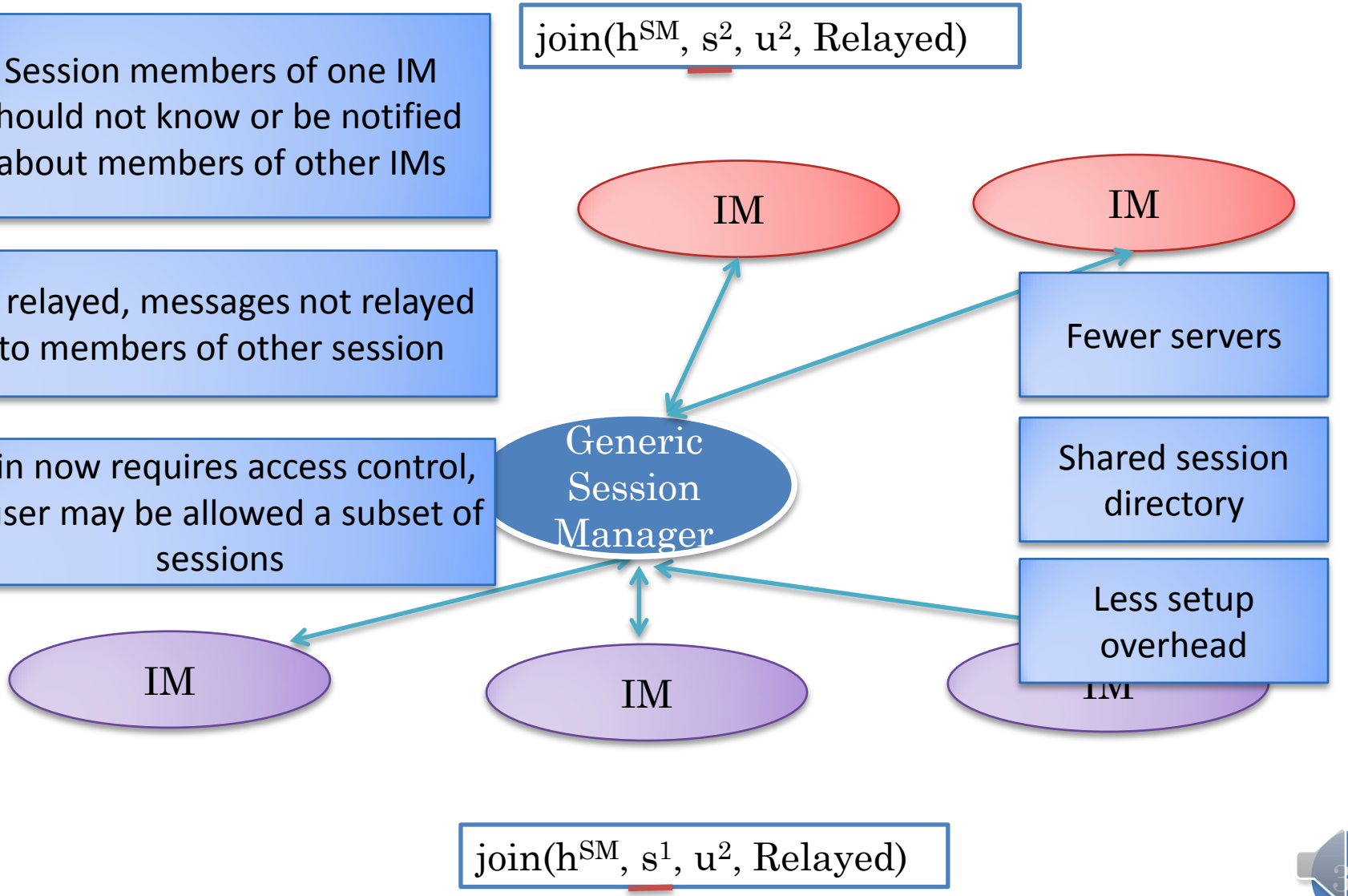


Multiple Sessions Per Session Manager

Session members of one IM should not know or be notified about members of other IMs

In relayed, messages not relayed to members of other session

Join now requires access control, a user may be allowed a subset of sessions



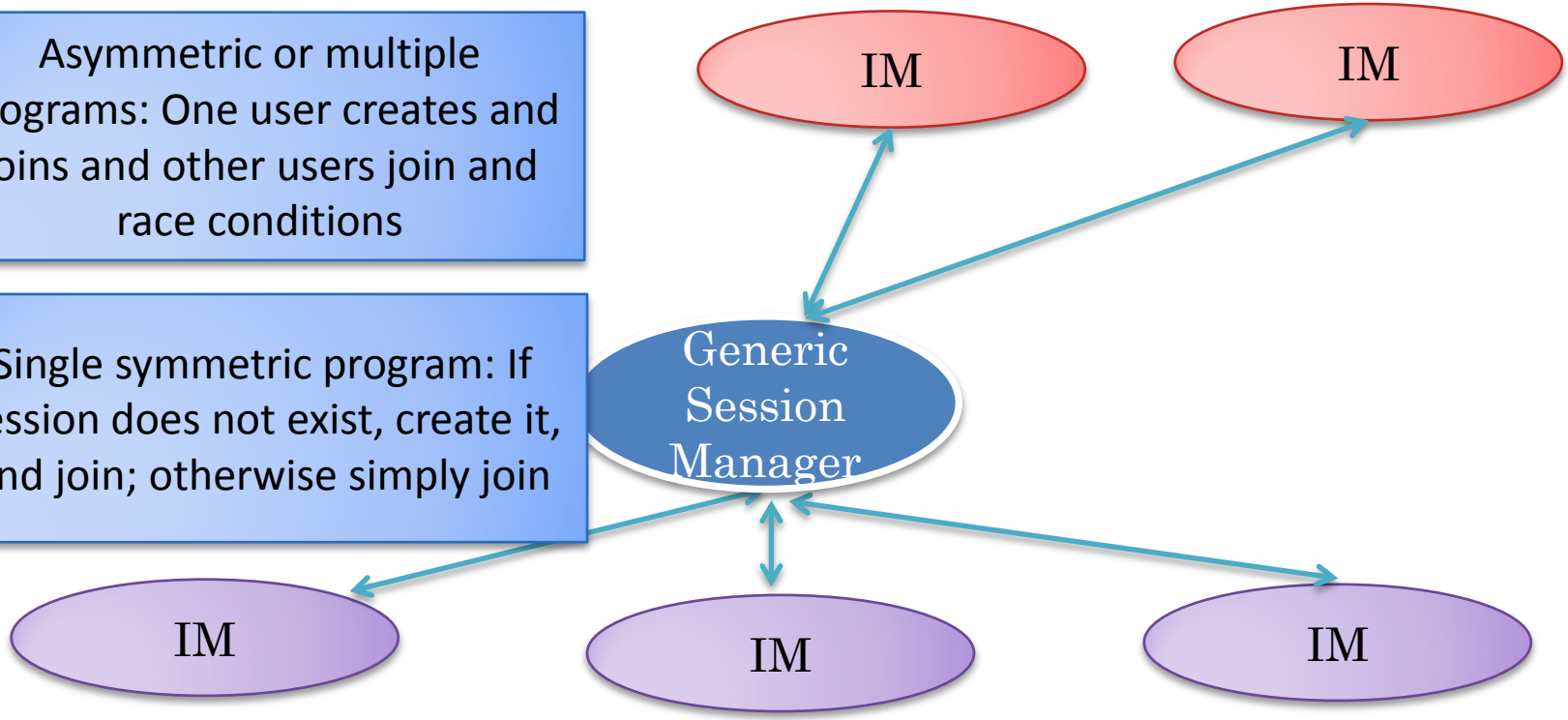
Asymmetric vs. Symmetric Joins

Create call?

join(h^{SM}, s^2, u^2, Relayed)

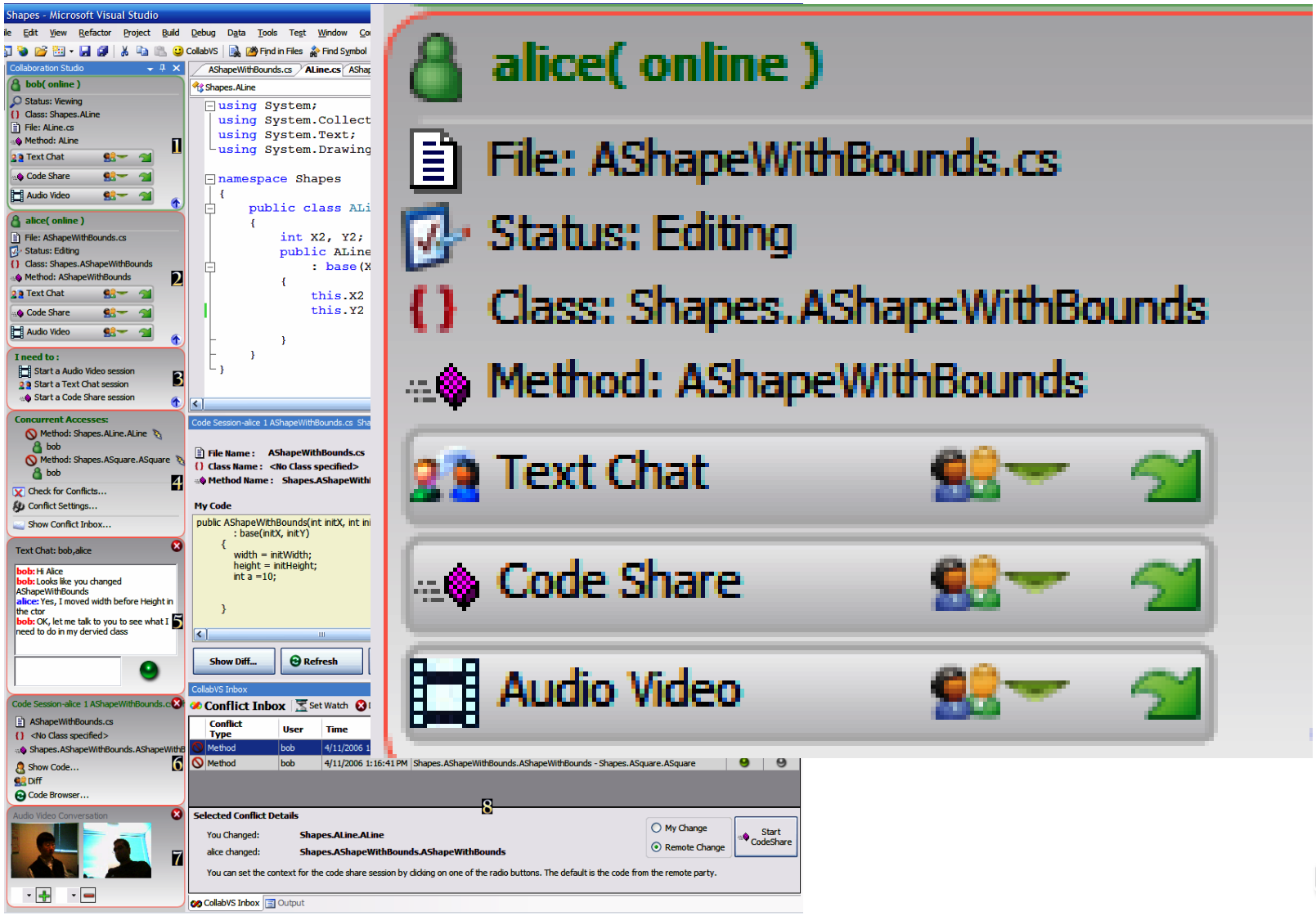
Asymmetric or multiple programs: One user creates and joins and other users join and race conditions

Single symmetric program: If session does not exist, create it, and join; otherwise simply join

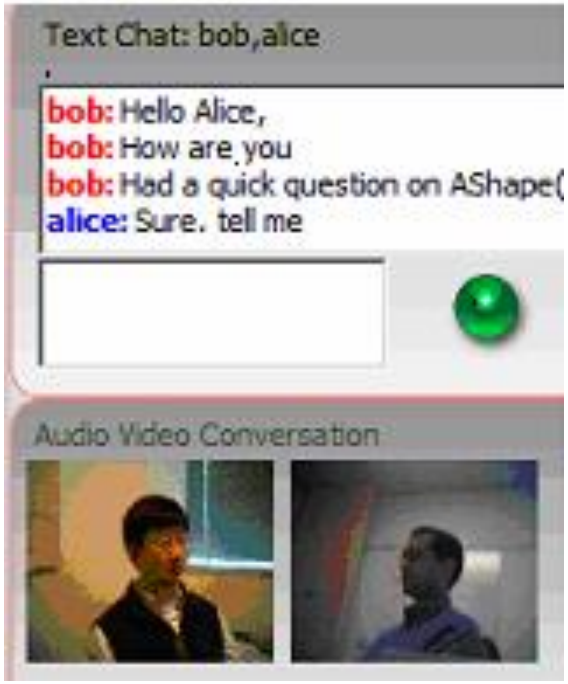


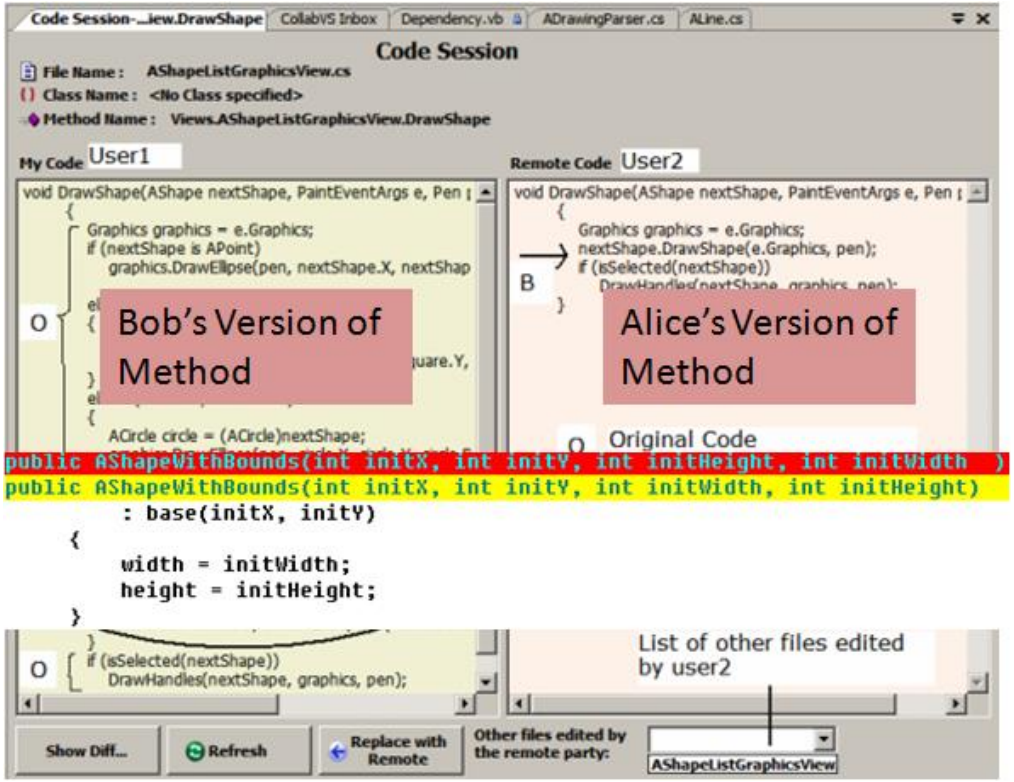
join(h^{SM}, s^1, u^2, Relayed)

Multiple Sessions?



Multiple Sessions?

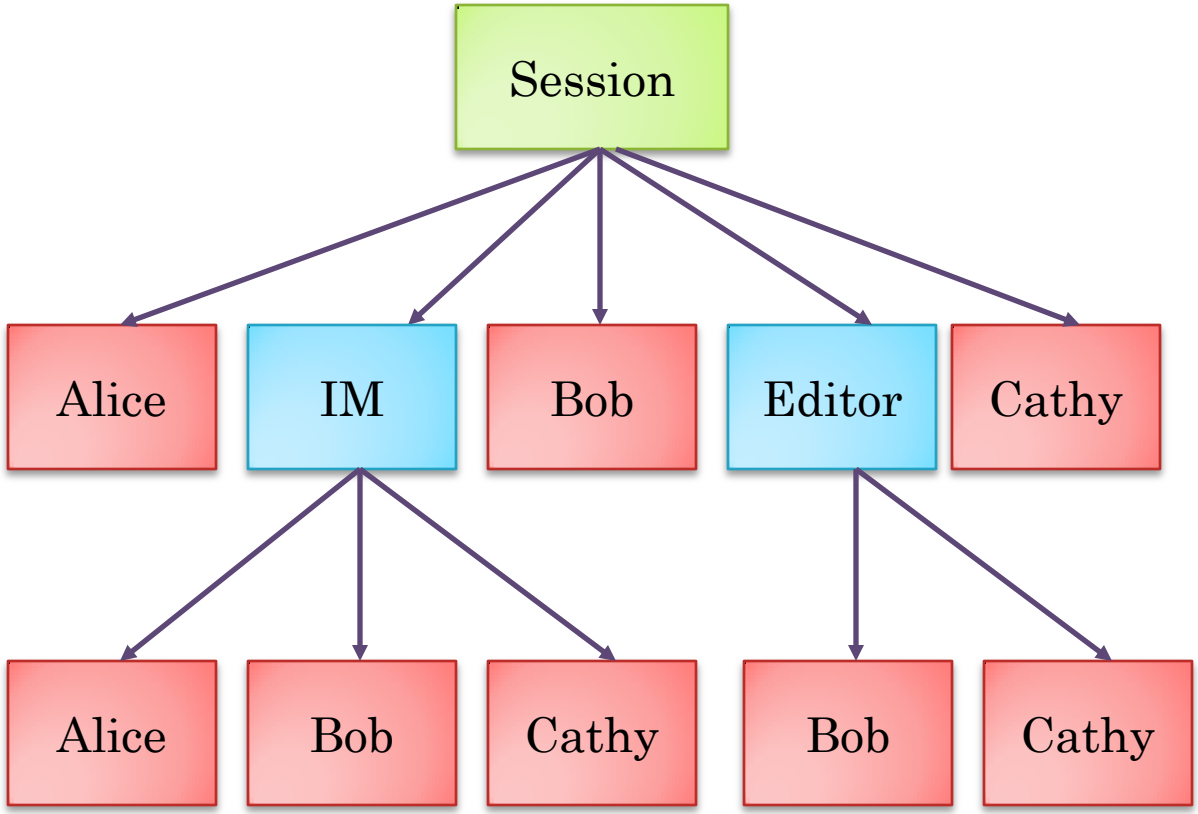




Are text chat, code share and audio video separate sessions?

They are different application sub sessions in same session, no access control required to join applications once user is in session, and each user knows about users of each application

Session with Application Sub-Sessions

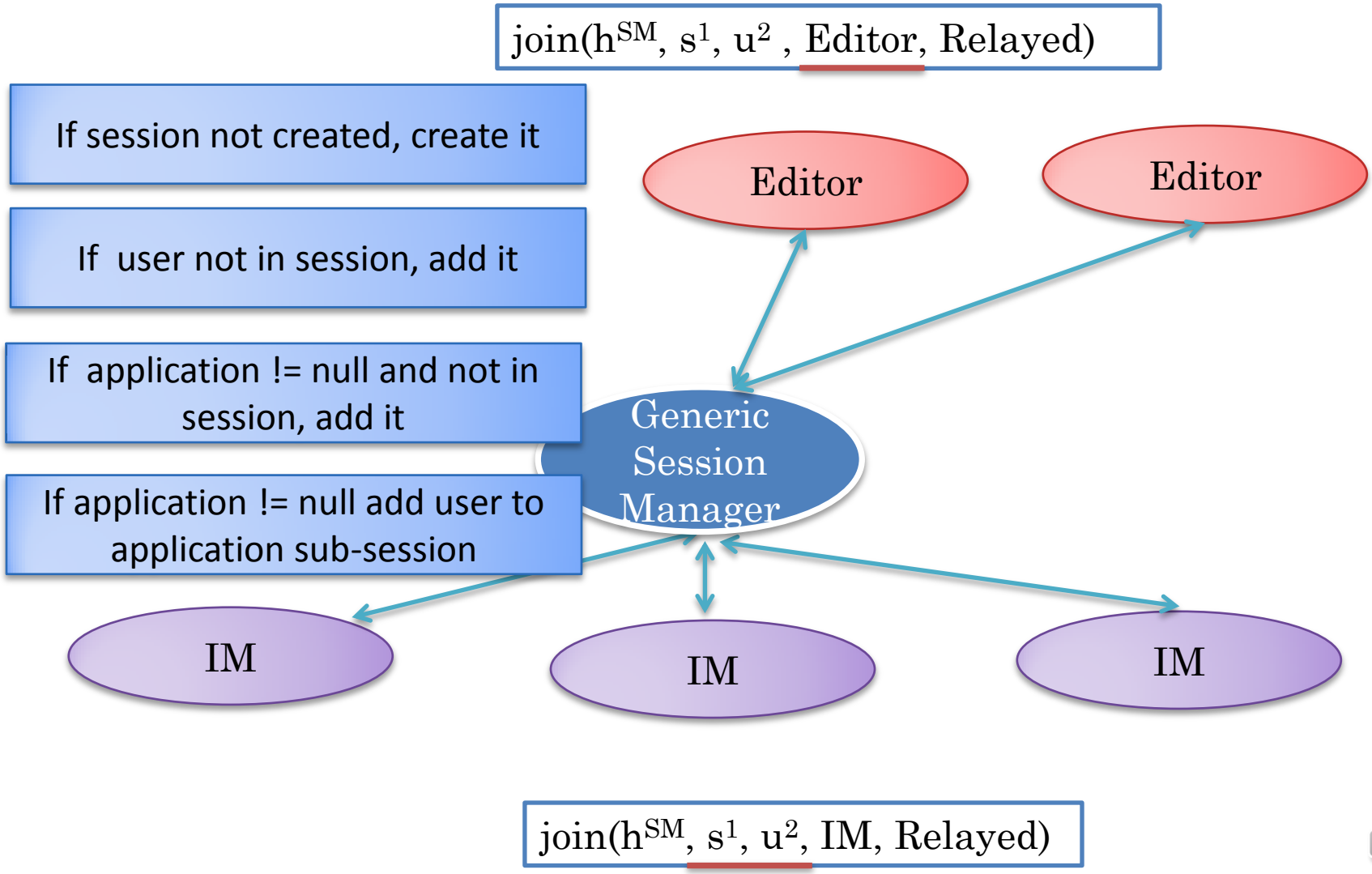


Users in session can interact with any application in session – access control done when user joins session

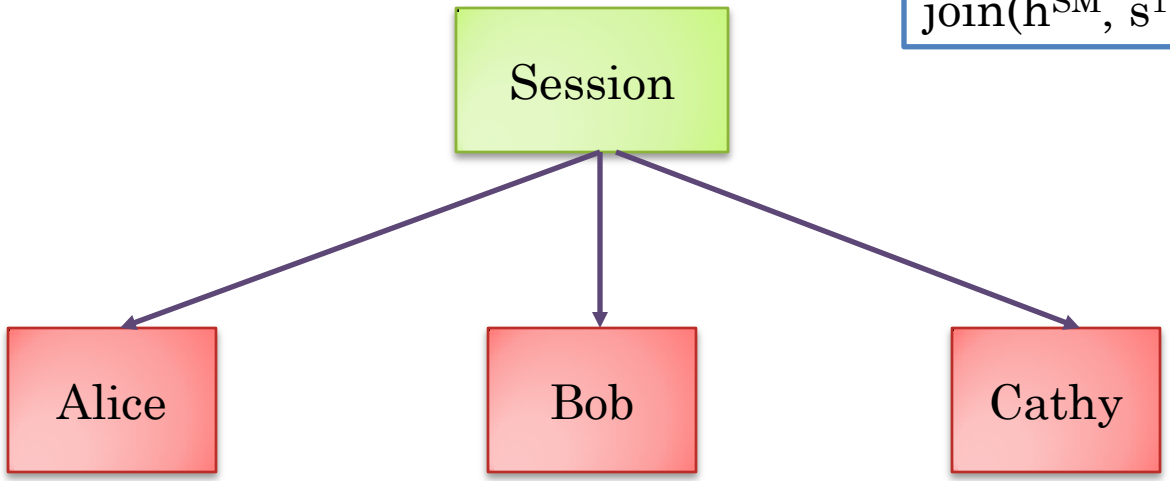
User notified of all applications added to session and all users interacting with the application

Need to keep users of applications separate so that multicast calls can distribute messages correctly, so users in application sub-sessions

Multiple Applications per Sessions



Flat Sessions (Review)



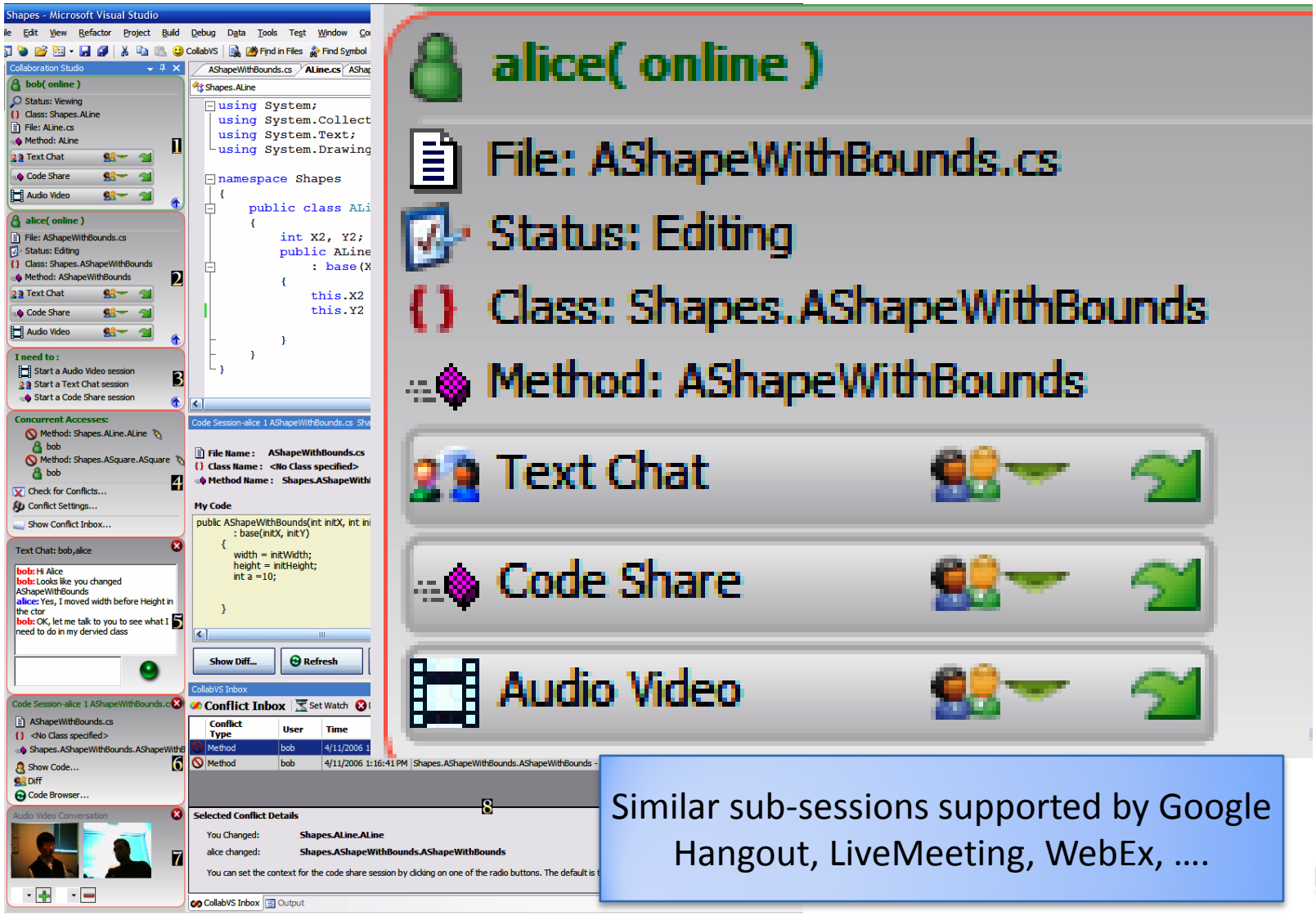
join(h^{SM}, s^1, Cathy, Relayed)

Some process running on behalf of Cathy joins the session

Session manager oblivious of application implemented by process

Session manager oblivious of application implemented by process

Motivating Sub-sessions

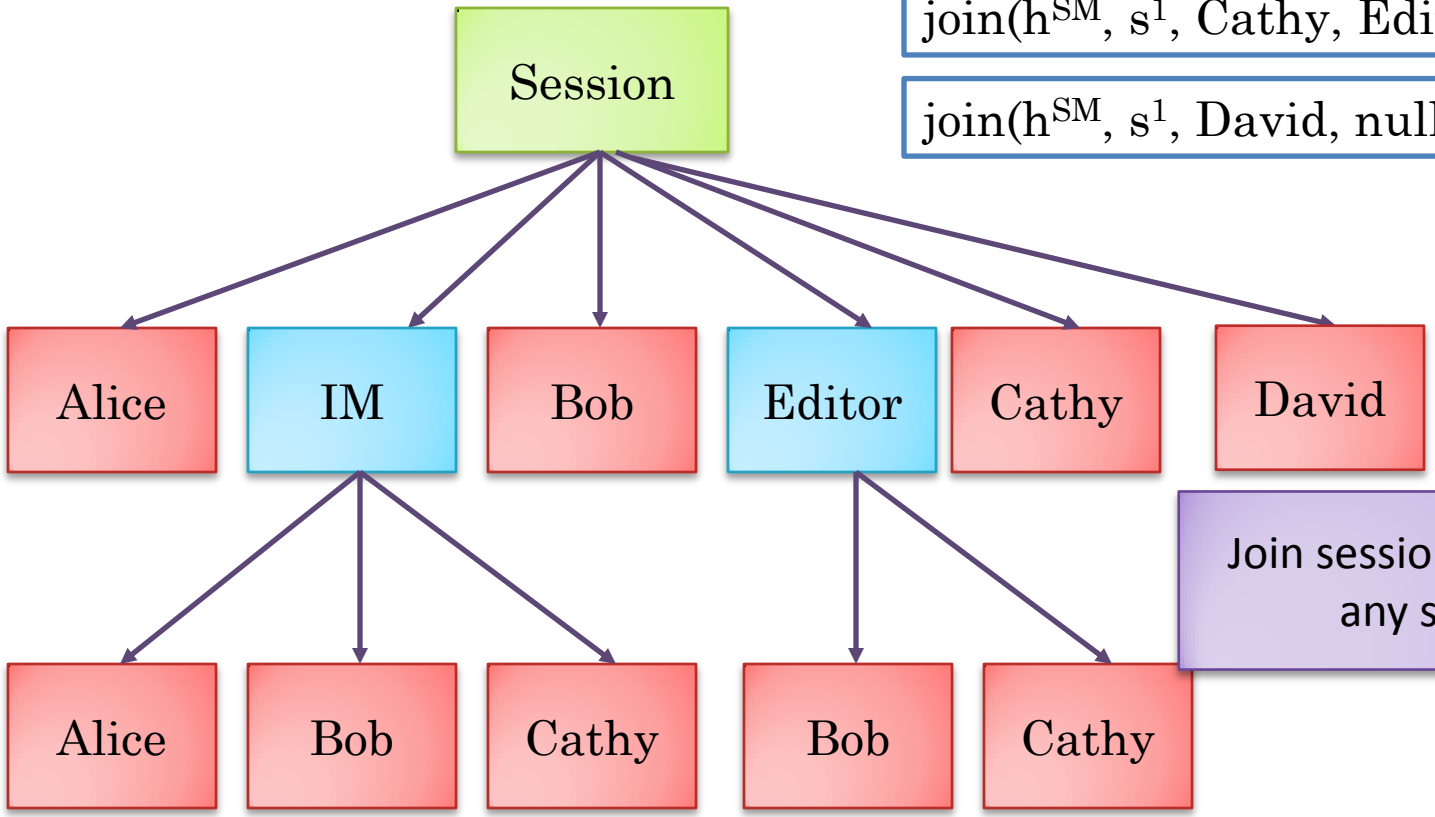


Session with Application Sub-Sessions

join(h^{SM}, s^1, Cathy, IM, Relayed)

join(h^{SM}, s^1, Cathy, Editor, Relayed)

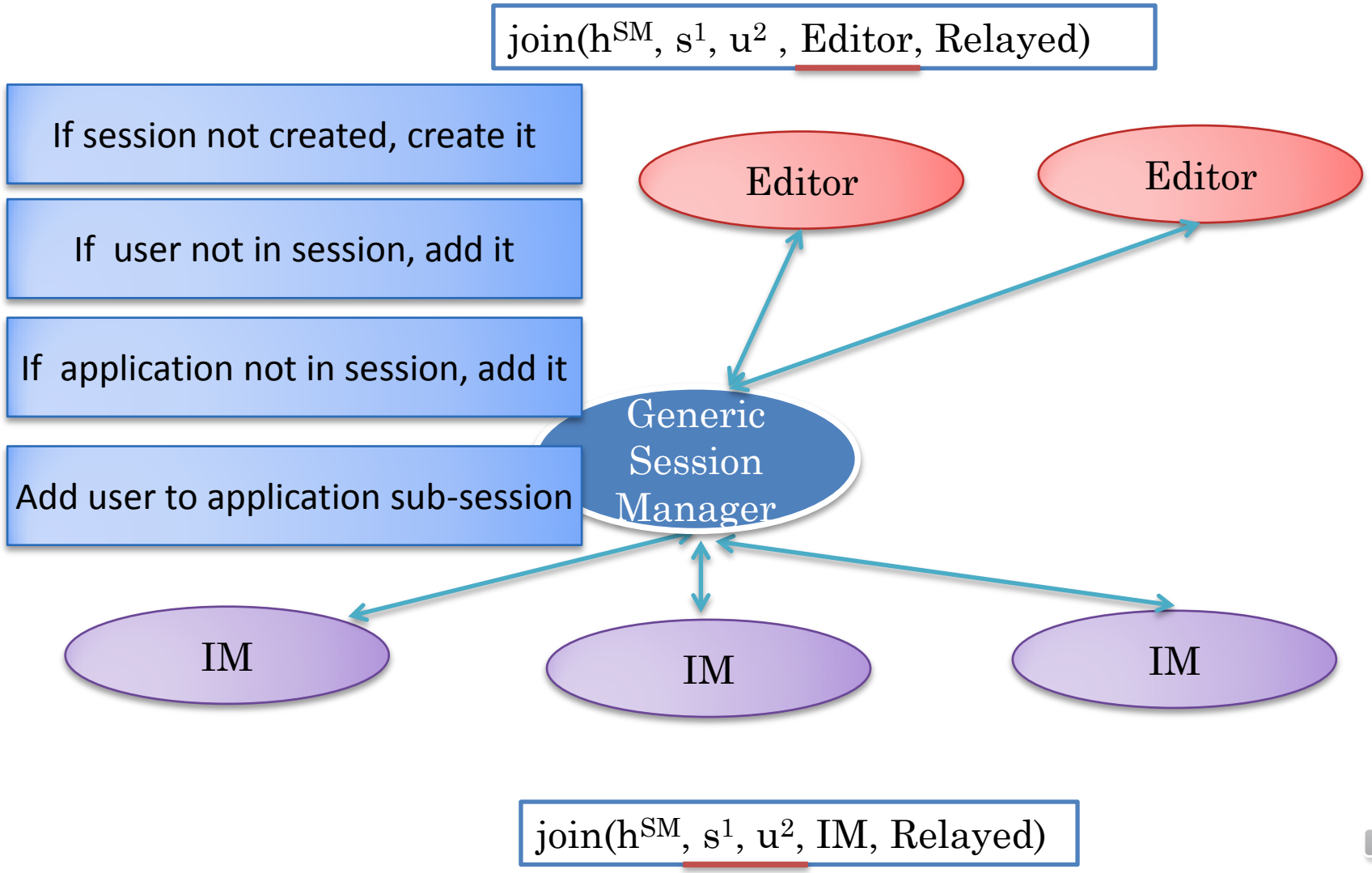
join(h^{SM}, s^1, David, null, Relayed)



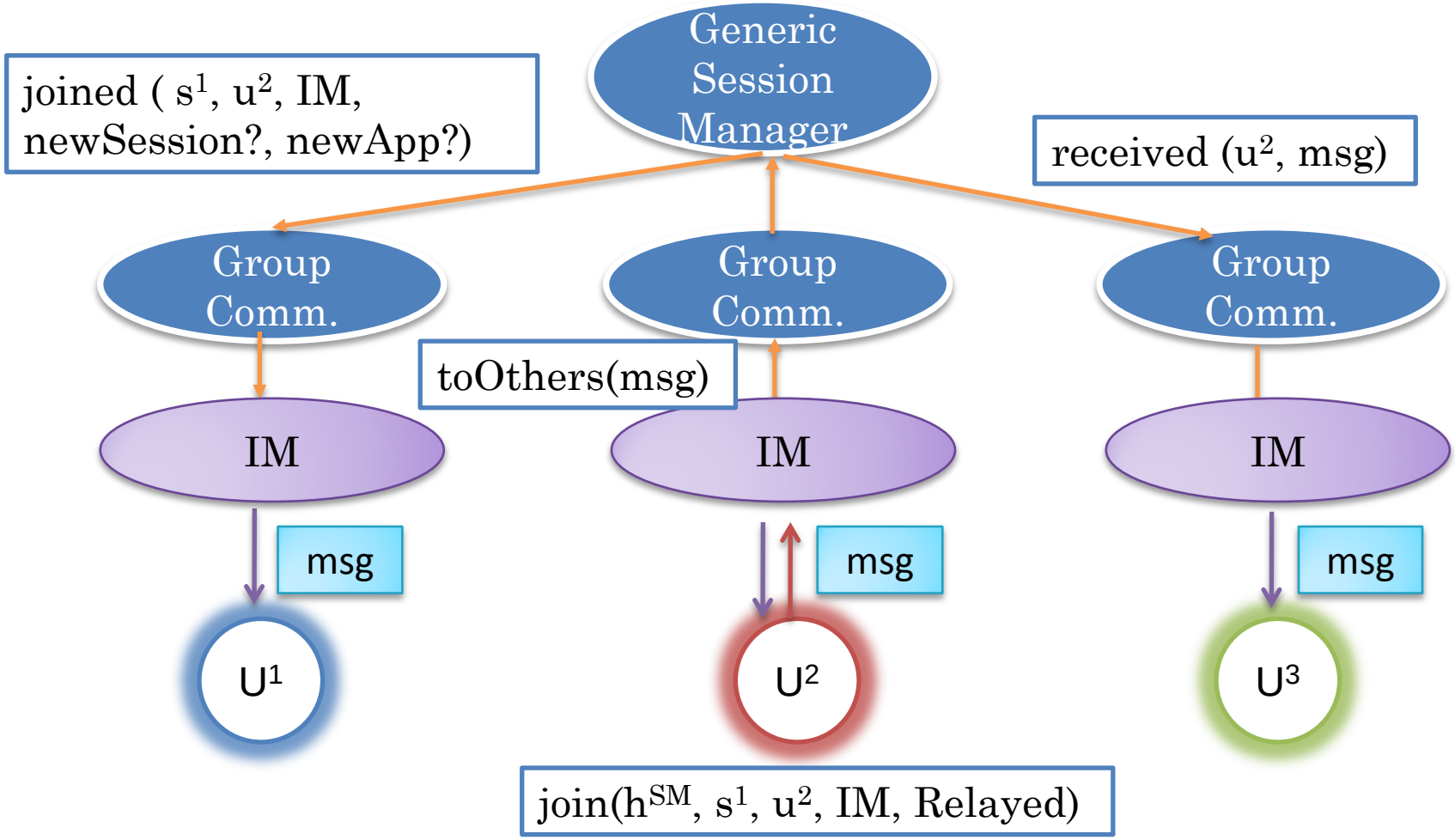
Join session without joining any subsession?

David's process cannot send or receive messages and simply listens to session callbacks, which can inform its user of session activity and join application sub-sessions

Session Join Semantics



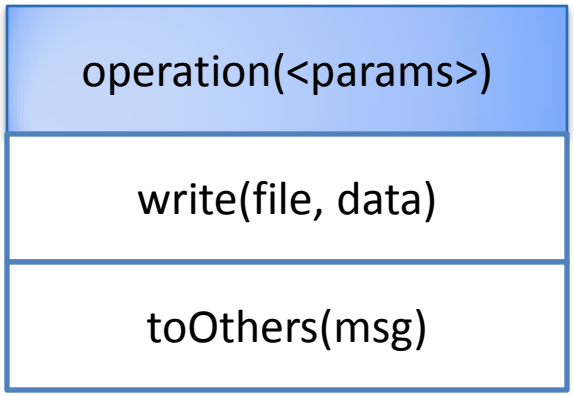
Synchronization



Semantics specified?

When do these calls/callbacks return: Synchronous vs. asynchronous?

Synchronous vs. Asynchronous

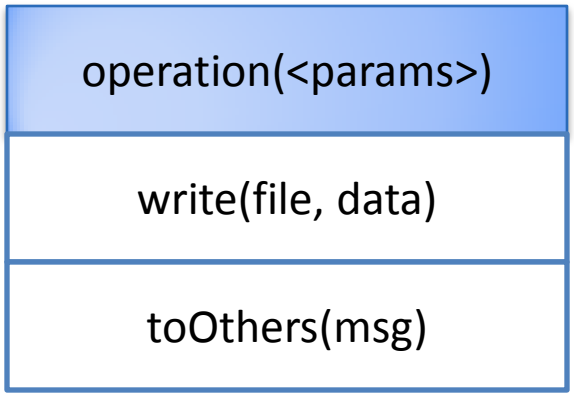


Synchronous: Operation invoker waits until the operation "finishes"

Asynchronous: Operation invoker does not wait until completion

Some other operation (e.g. callback) needed to wait for result or completion status

Synchronous vs. Asynchronous vs. Blocking Operations

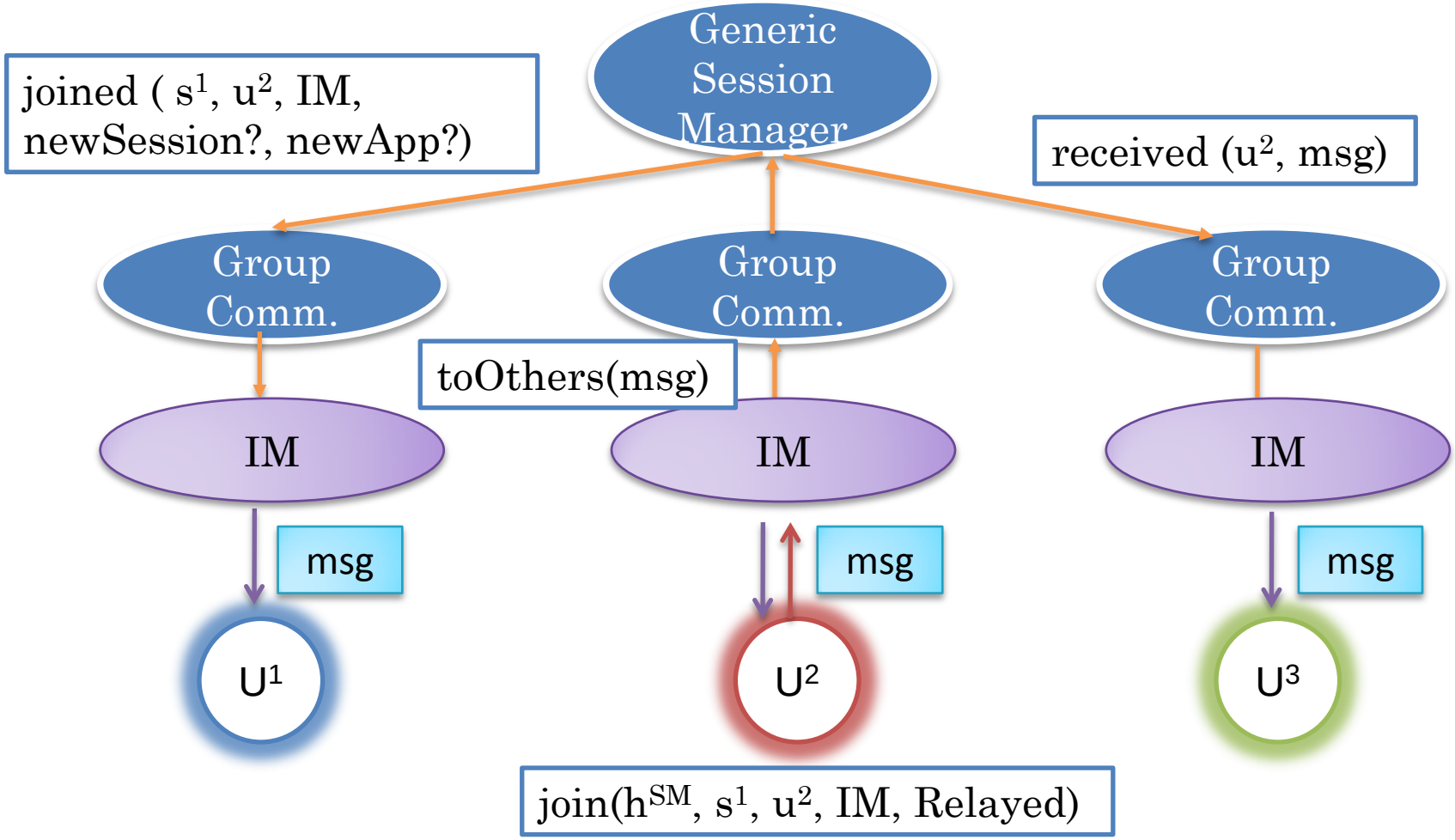


Blocking: Operation invoker waits, unblocking possibly before, until, or after operation completion (e.g. when data given to local OS)

Synchronous is always blocking

Blocking is not always synchronous

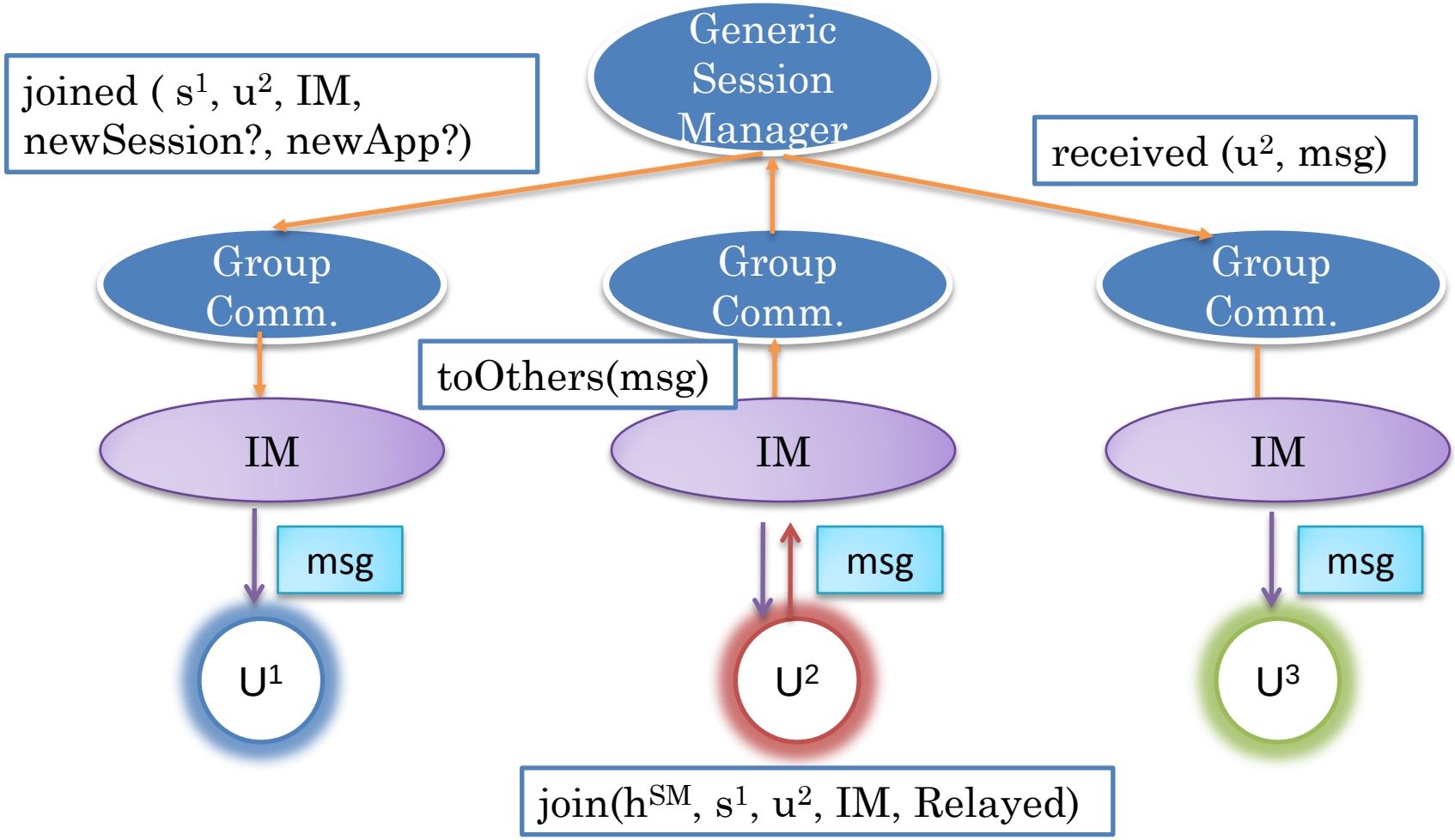
Operation Completion: Calls vs. Callbacks



Callback finishes when it finishes execution (at the other site)

Call finishes when the call and any associated callbacks finish

Calls and Callbacks: Sync vs Async?



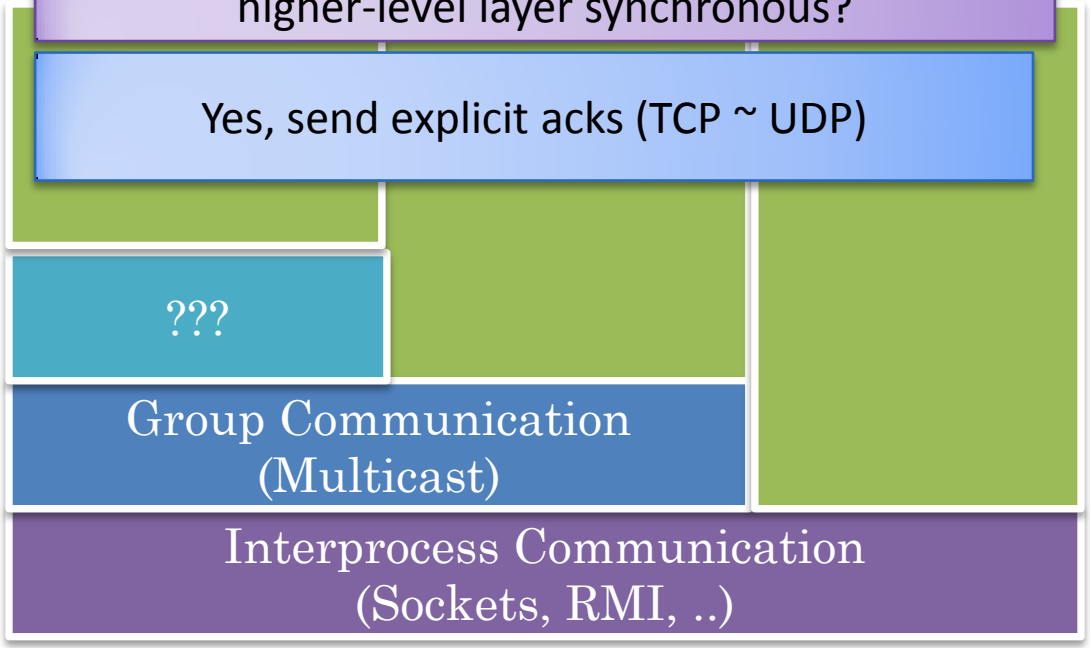
Synchronous call: Local response effected

Synchronous call/callback: Local and remote response affected

Inter-Layer Dependencies

If lower-level layer, is asynchronous can we make higher-level layer synchronous?

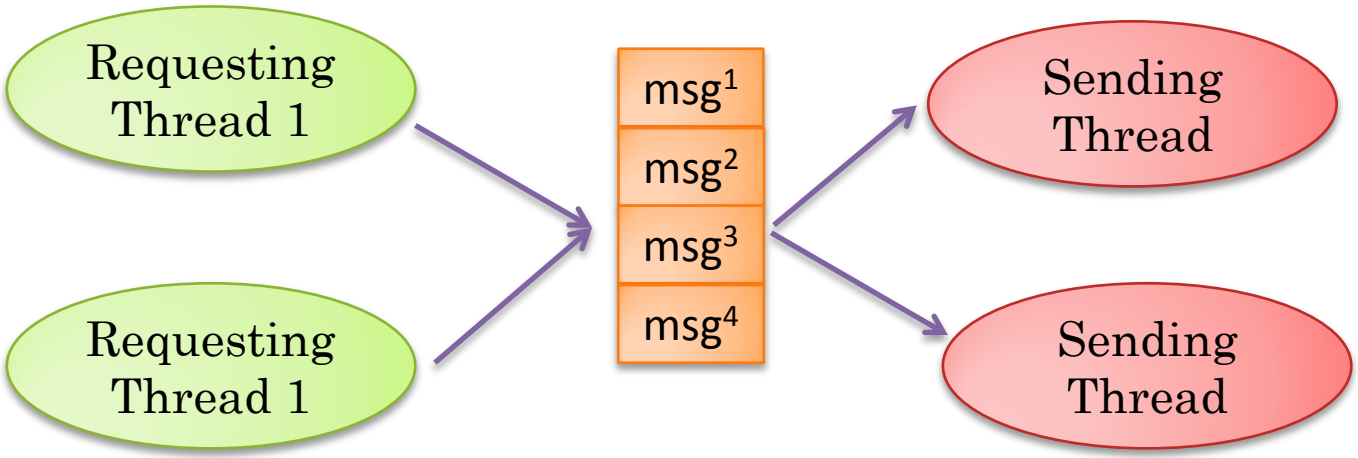
Yes, send explicit acks (TCP ~ UDP)



If lower-level layer, is synchronous can we make higher-level layer asynchronous?

"Yes", with separate threads

Multiple Threads



Producers

Bounded Buffer

Consumers

Requesting thread passes message to sending thread and does not block

Communication between the two threads example of some classical problem?

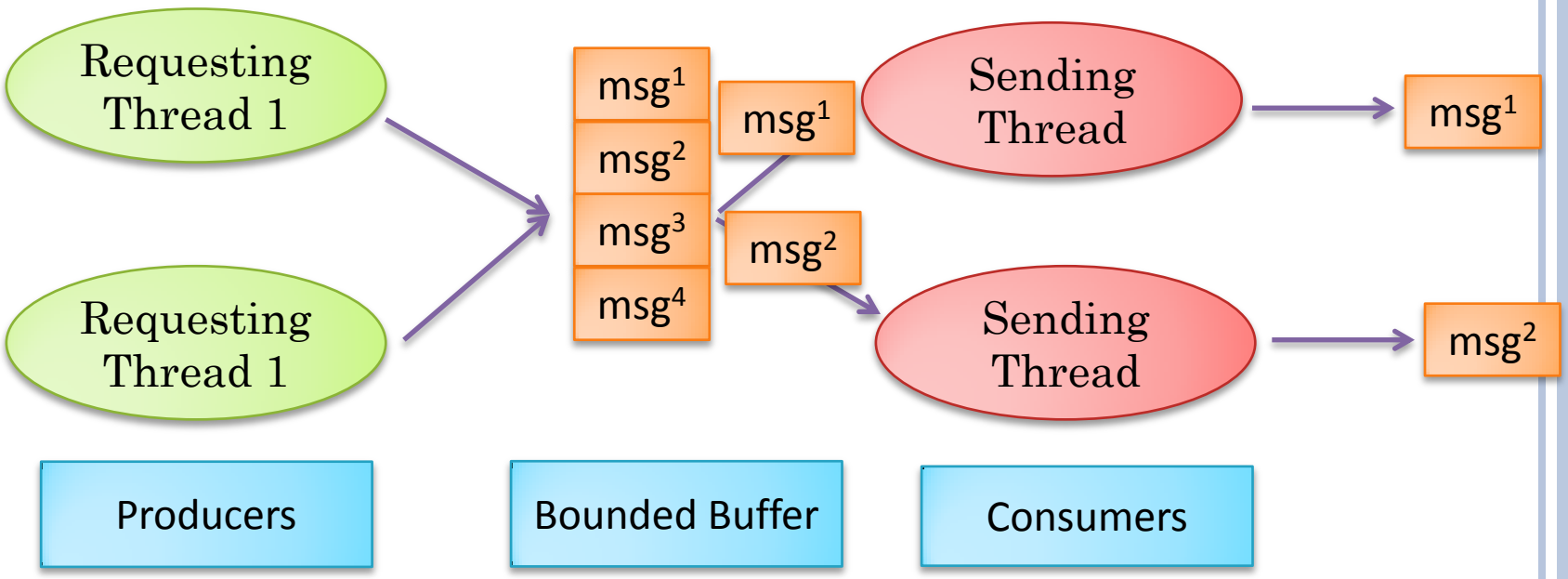
Sending thread invokes synchronous operation

How many consumers?

What if the two threads work at different rates (expected) – what if more than one pending message?

Consistency constraints?

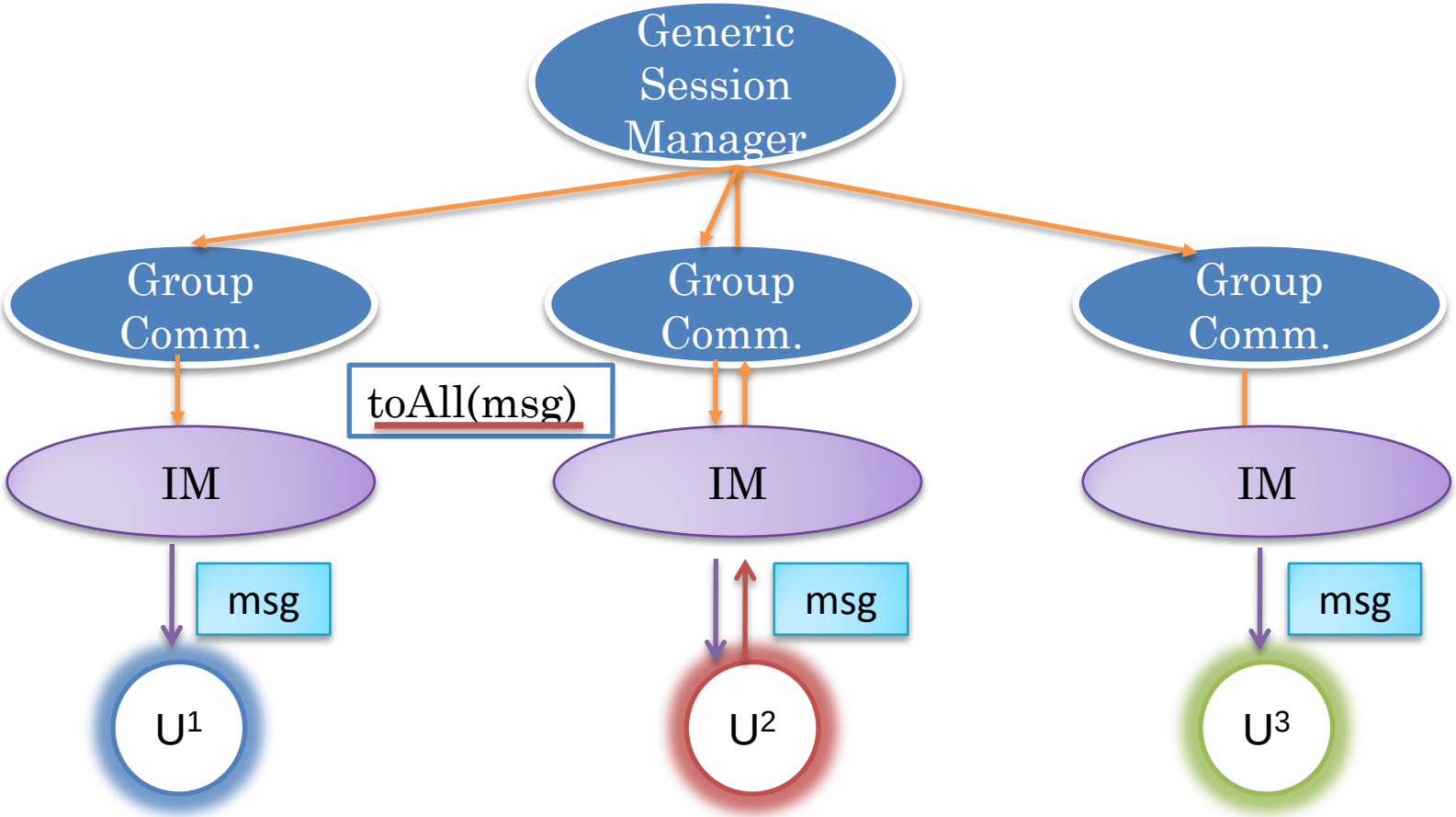
Multiple Threads



producers is application specific and cannot be controlled

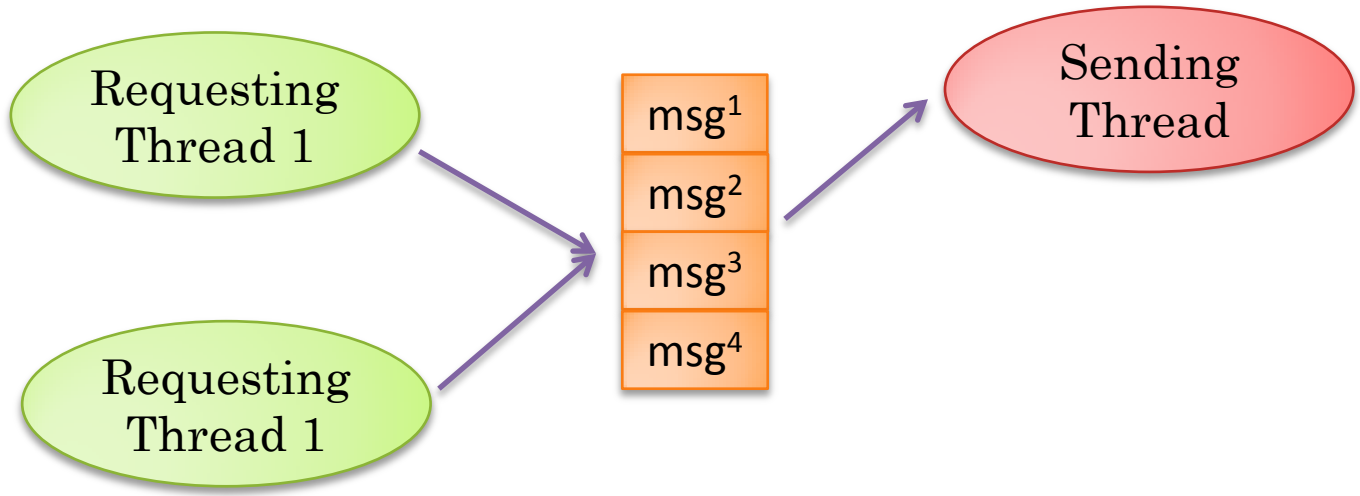
> 1 consumers can result in message reordering

Why Ordering is Important



Only messages within an application sub-session need to be ordered

How many Consumers



Asynchronous

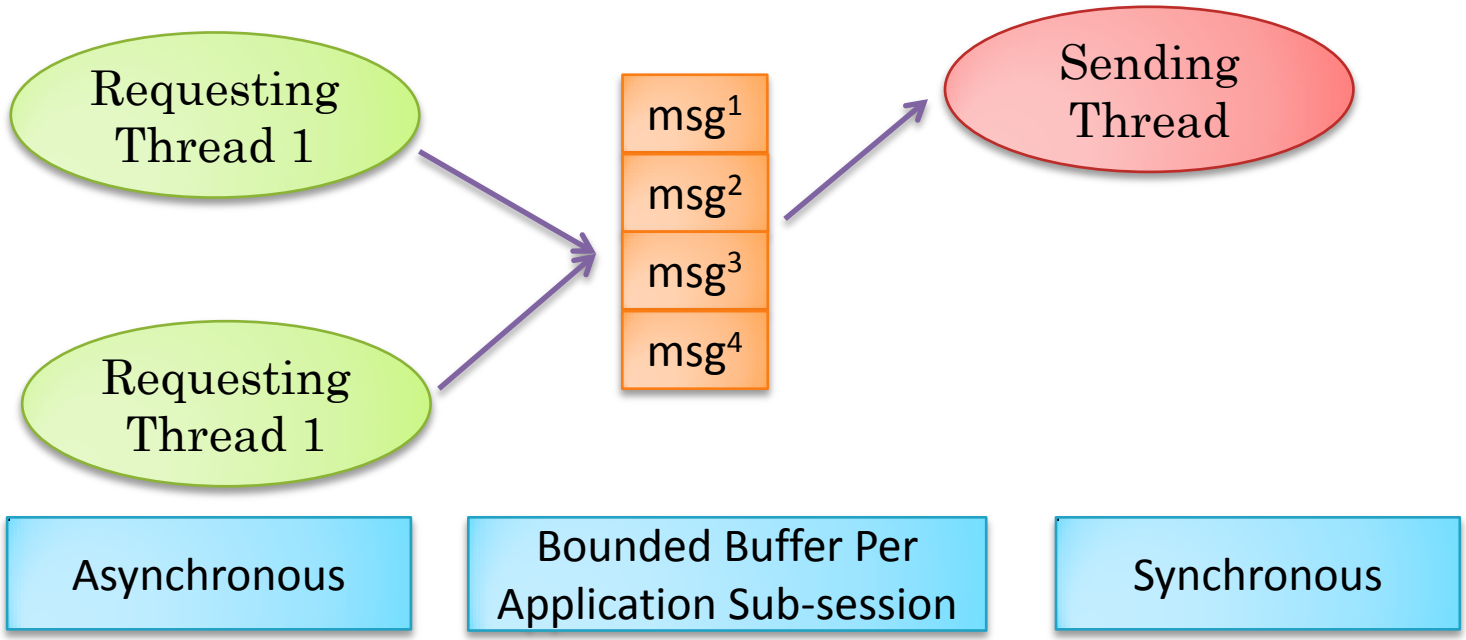
Bounded Buffer Per Application Sub-session

Synchronous

One sending thread per application sub session at both session server and its clients

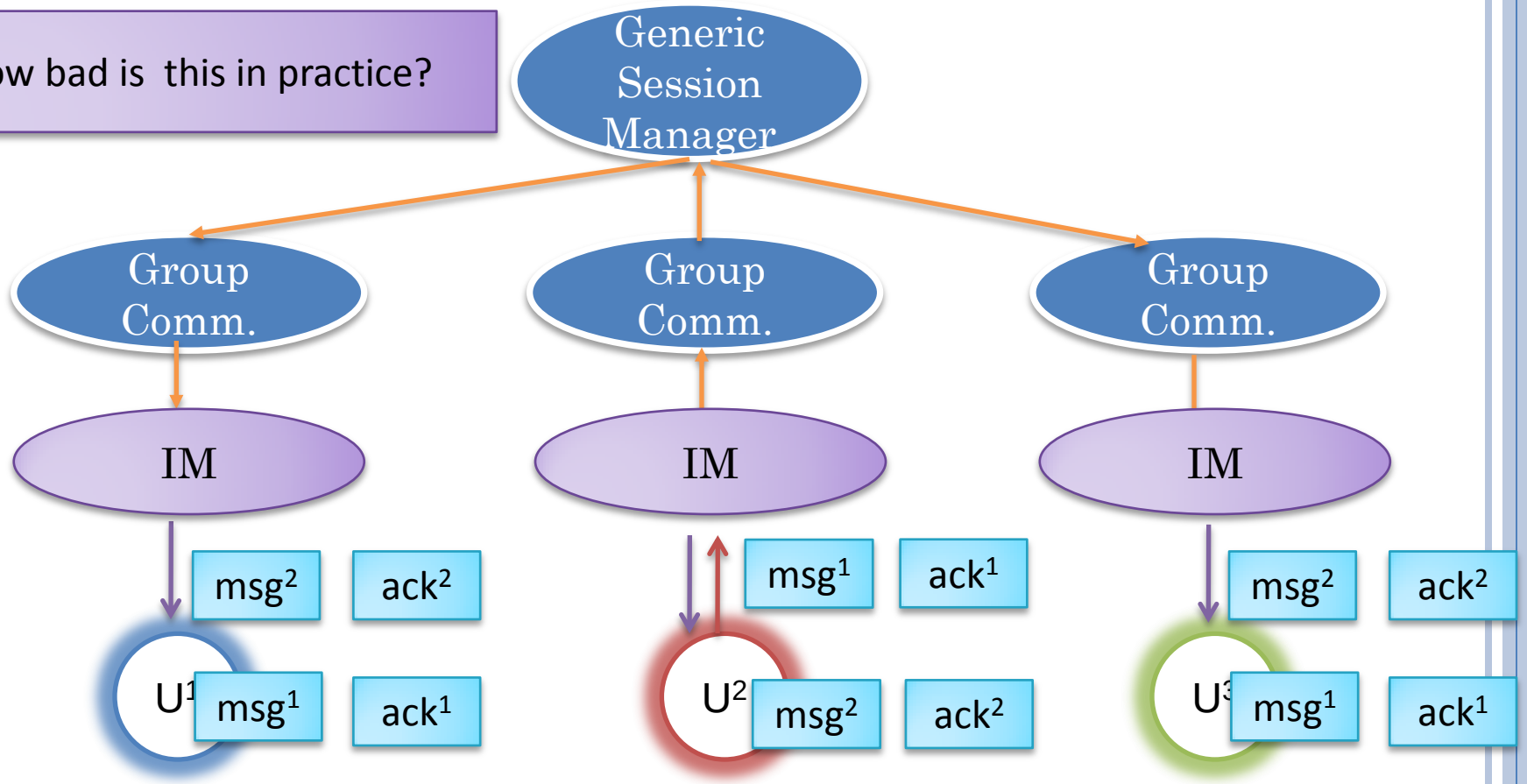
Multicast layer hides this from client

Feedback vs Feedthrough



Synchronous IPC

How bad is this in practice?

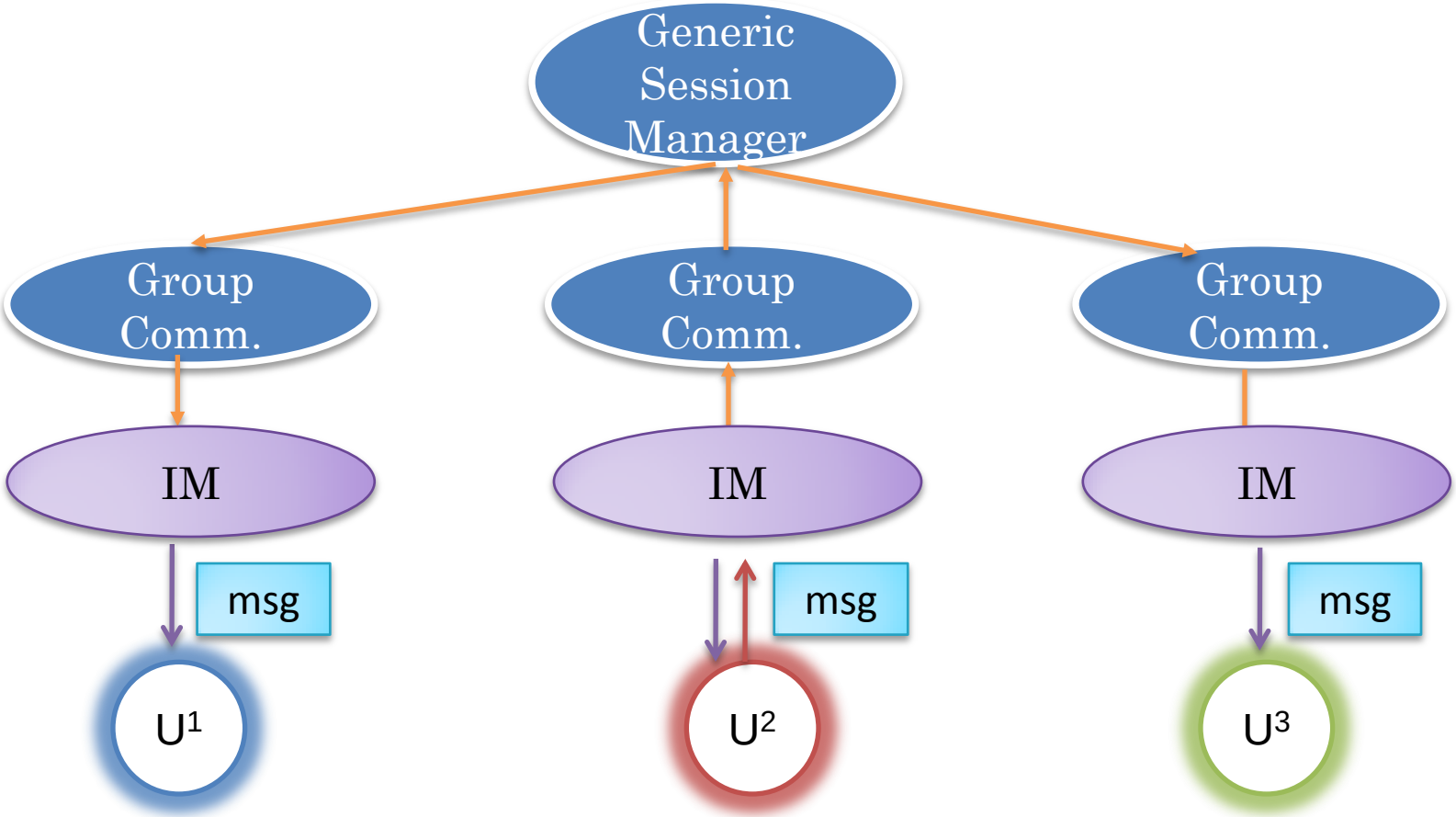


Round trip time ~ 100-2000ms

Time between telepointer movements: 10 ms

Noticeable (tolerable) latency: 30ms, 500ms

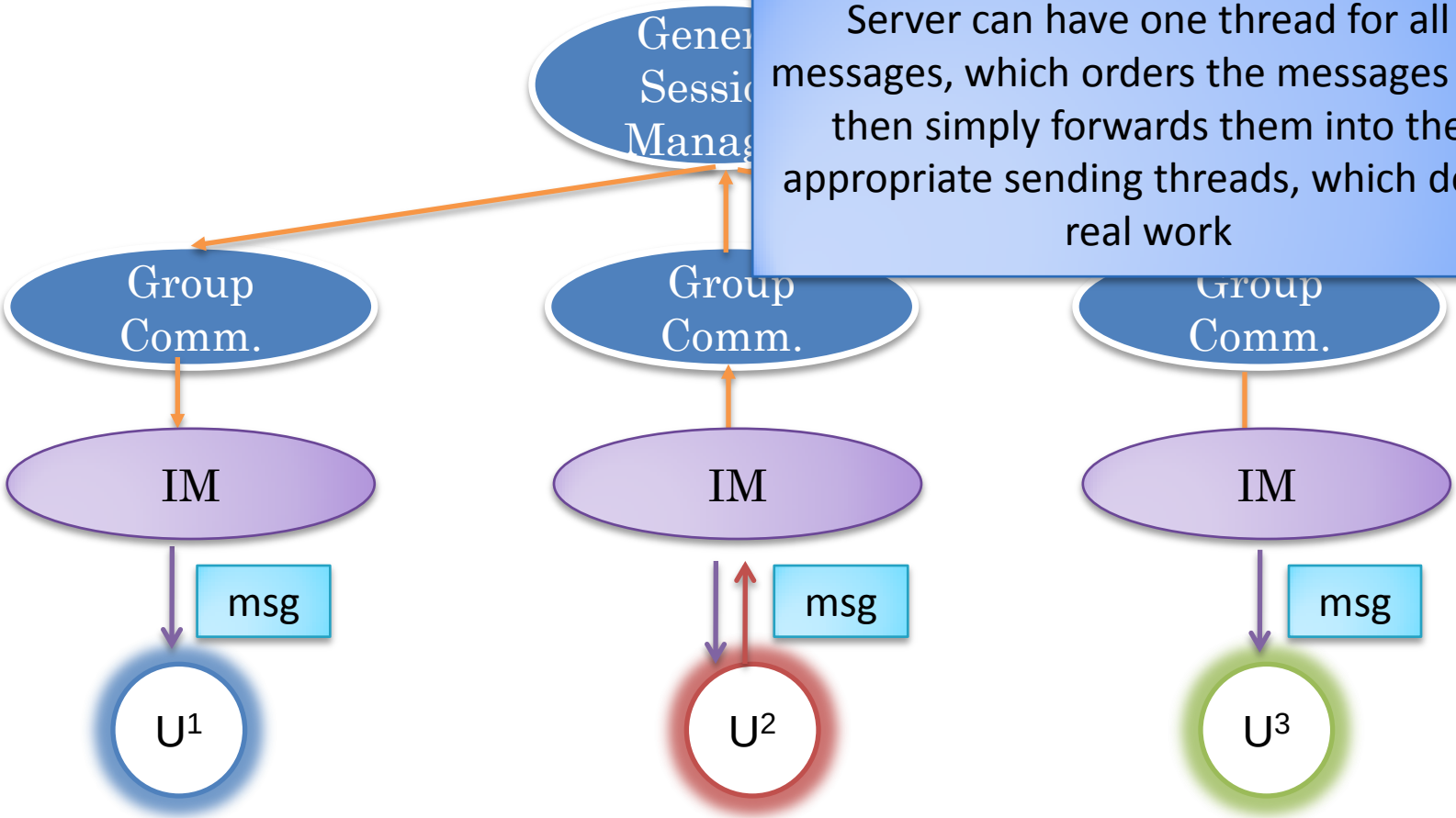
Message Receipts?



Threads that receive messages?

Group communication layer handles these threads

How many Receiving Threads?



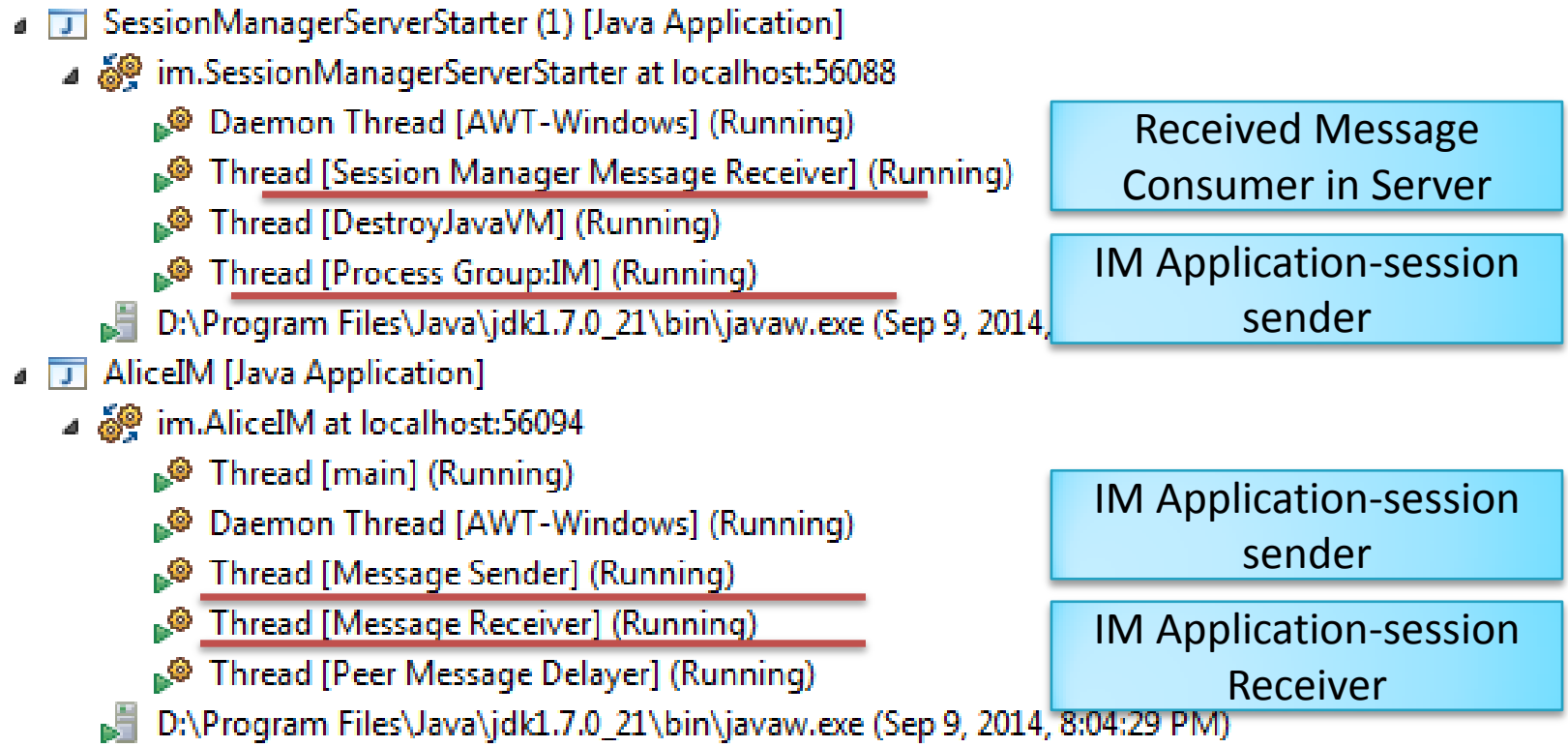
Server can have one thread for all messages, which orders the messages and then simply forwards them into the appropriate sending threads, which does real work

Messages received for an app subsession should be handled by one thread

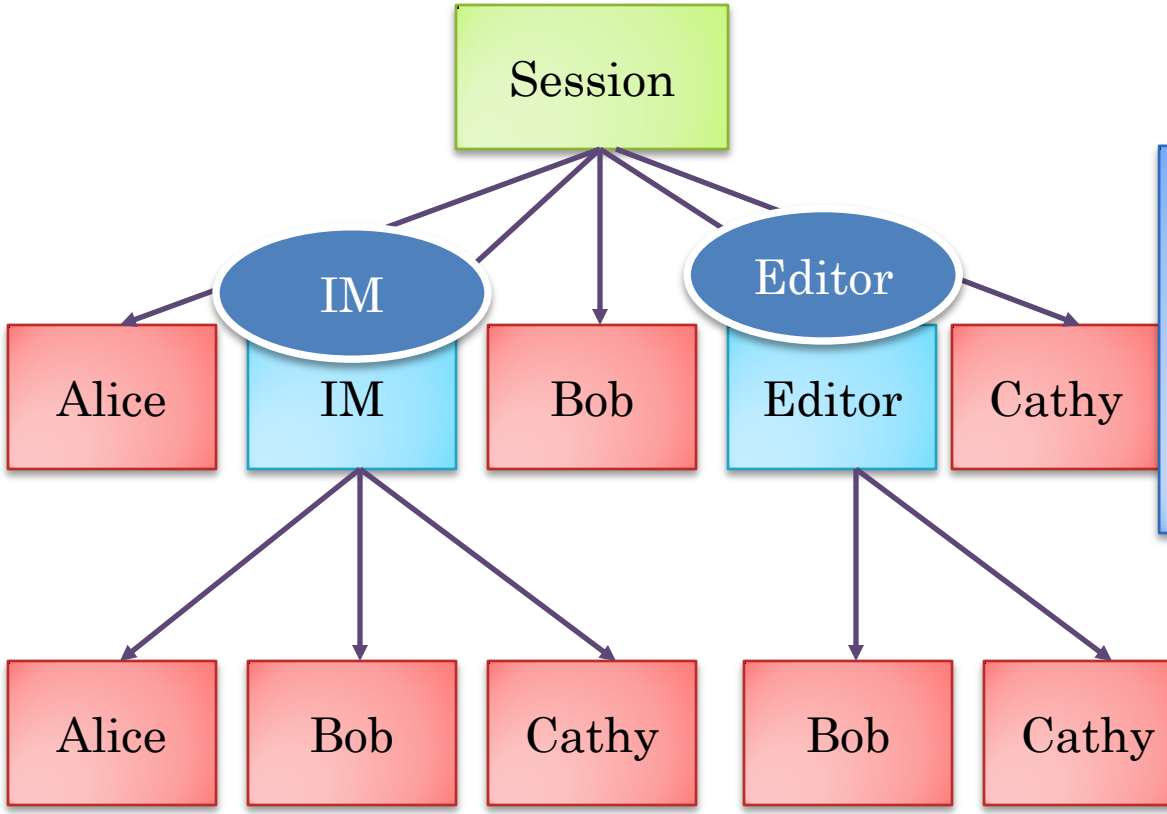
Client can have one thread per application subsession

How many client and server threads?

Threads (Concrete Example)

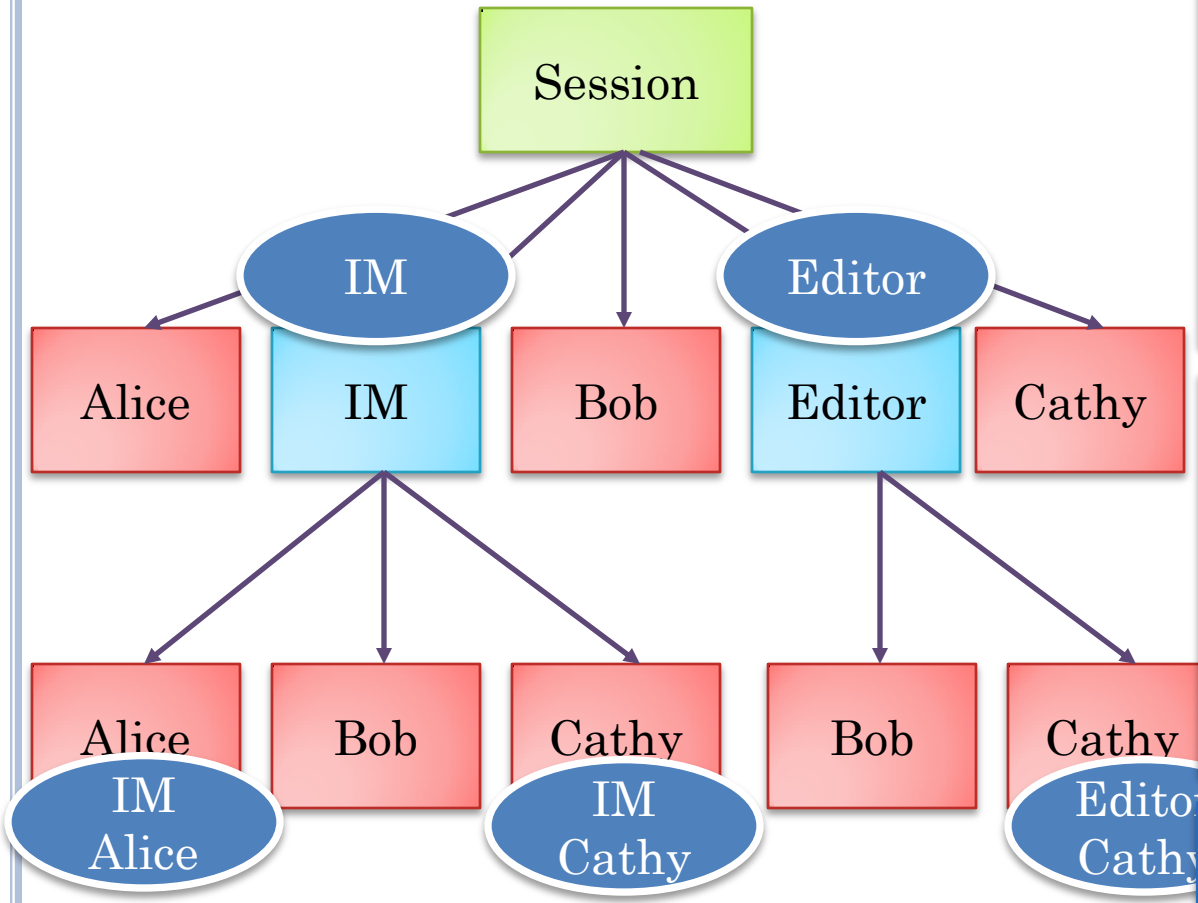


Sending Threads: BOB Client (Relayed)



Application-session threads send join and leave requests to session manager and also relay messages to session manager, waiting if necessary based on delay parameters

Sending Threads: Bob Client (P2P)

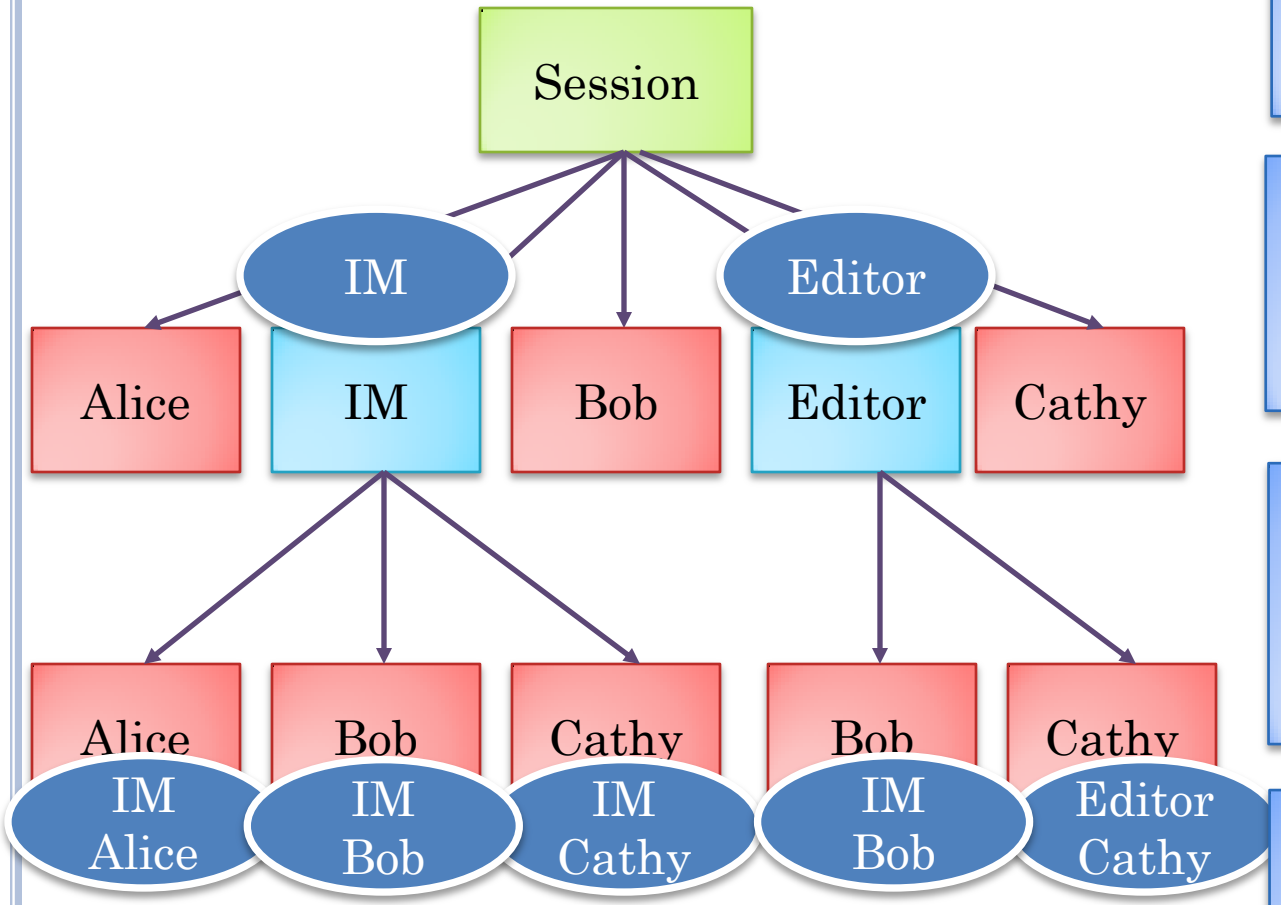


Application-session sending threads send join and leave requests to session manager and also serialize messages of the session, forwarding them to peer user threads

Peer sending threads receive messages from their application-session threads and send messages to peers, delaying messages based on delay parameters

Could share peer threads among application-session threads but more modular to create new threads

Sending Threads: Session Server



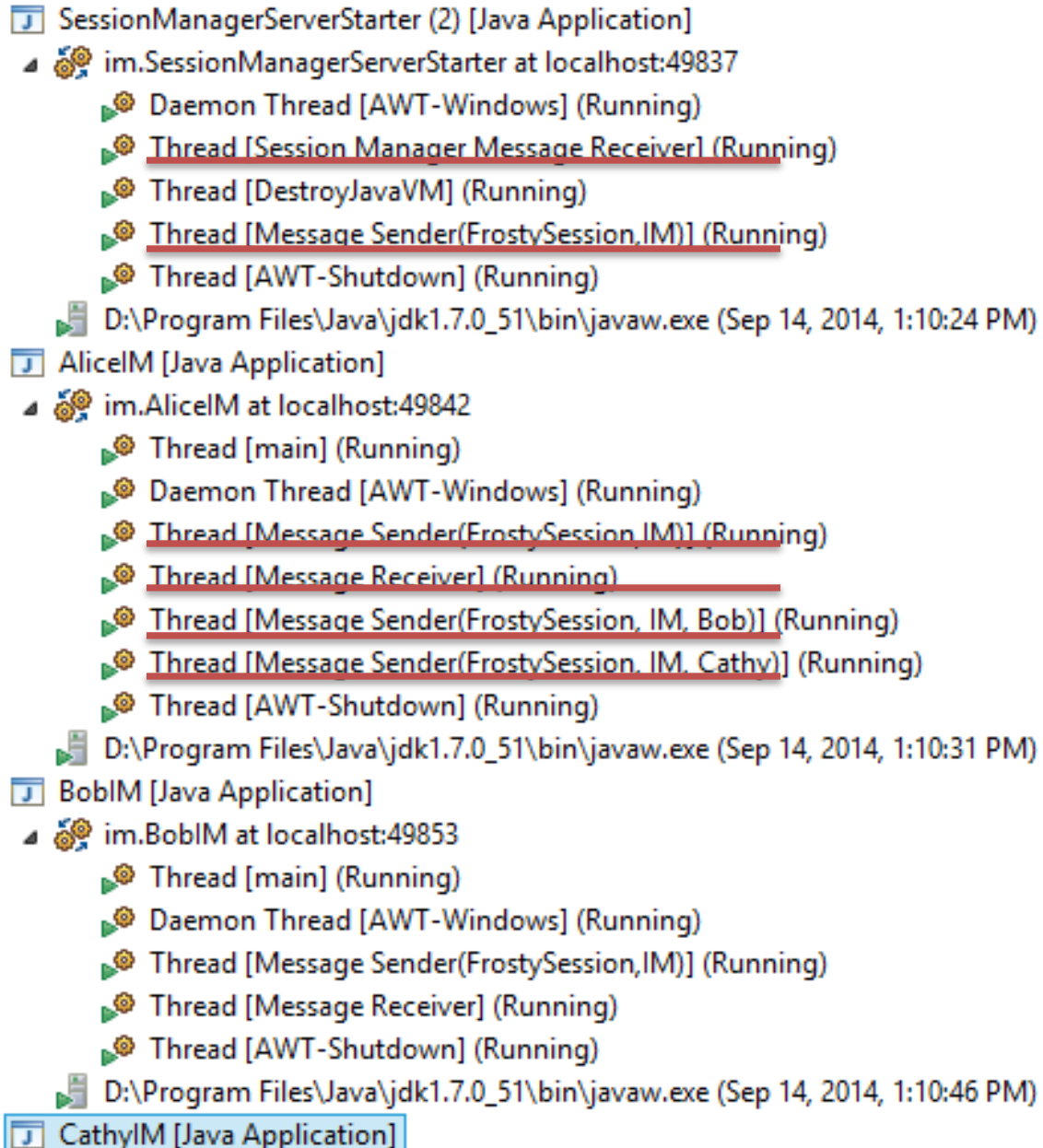
My implementation does not have per user thread at server

If feedthrough is an issue, use direct communication

A server may have numerous sessions, so per user thread maybe too much overhead

Moral: in production version do not use blocking IPC such as RMI

Concrete Threads (New Version)



Received Message Consumer in Server

IM Application-session sender

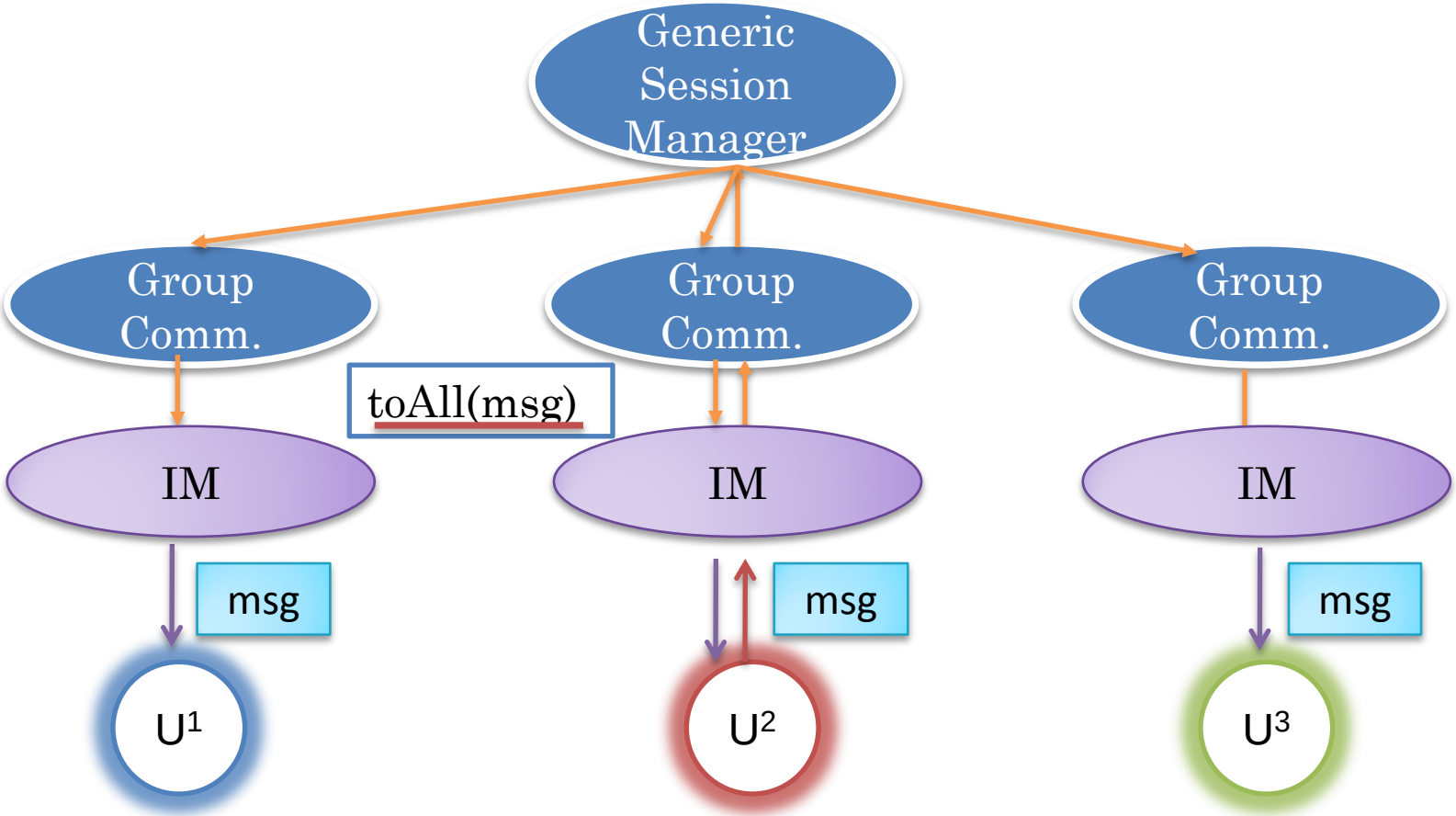
IM Application-session Receiver

IM Application-session Sender

IM Application-session, Peer Senders

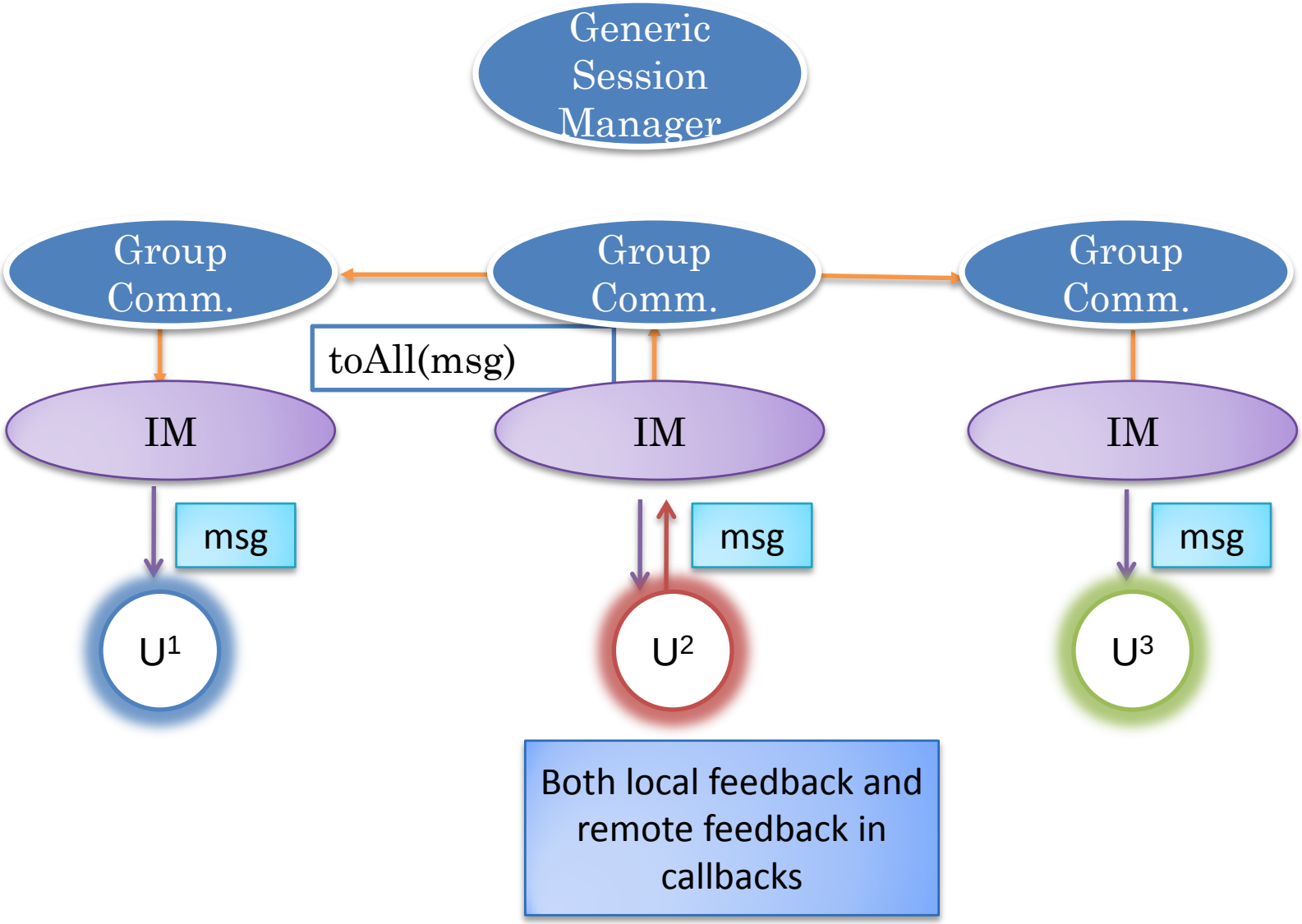
No peer senders for Bob as it is using relayed communication

toAll (Relayed)

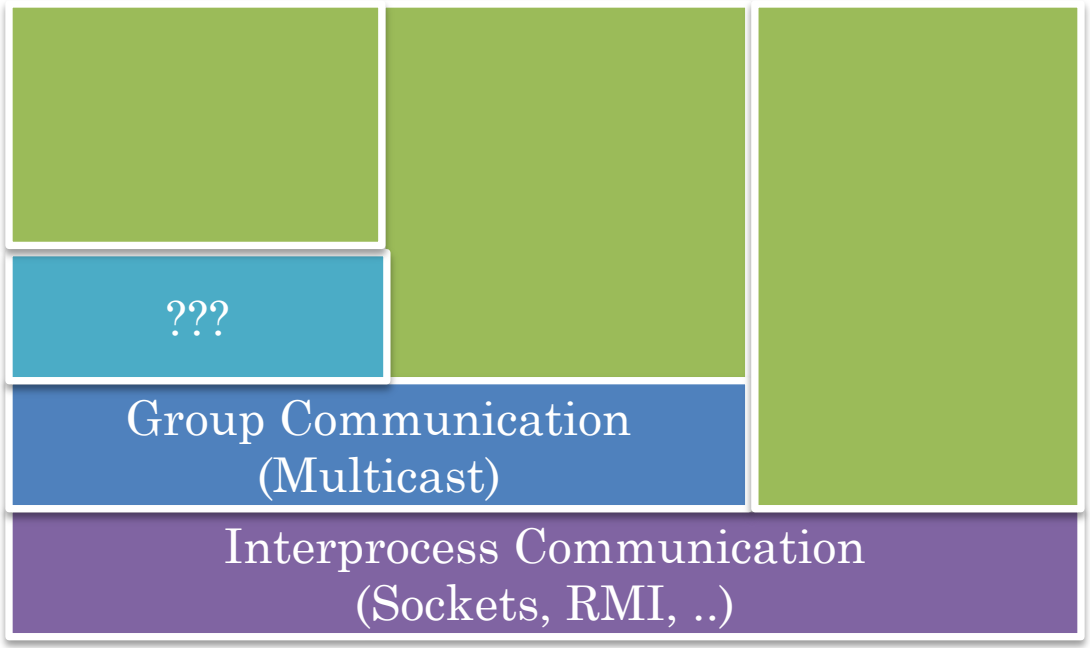


Do we need toAll() if messages are not relayed?

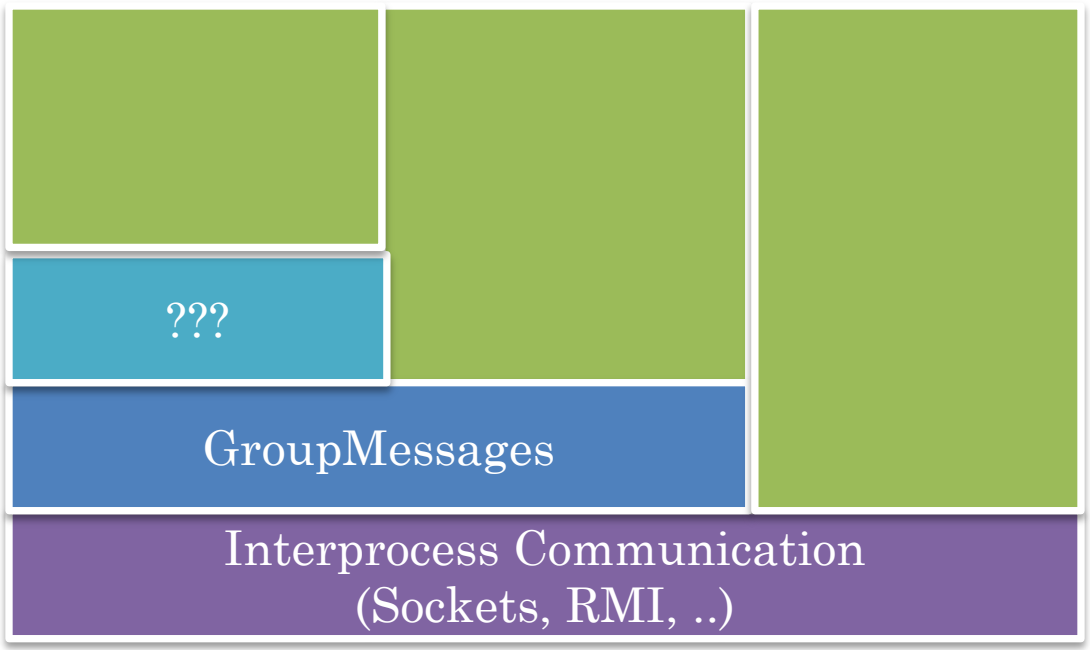
toAll(P2)


Generic Session Manager
Group Comm.
Group Comm.
Group Comm.
toAll(msg)
IM
IM
IM
msg
msg
msg
U1
U2
U3
Both local feedback and remote feedback in callbacks

The Concept of Group Communication

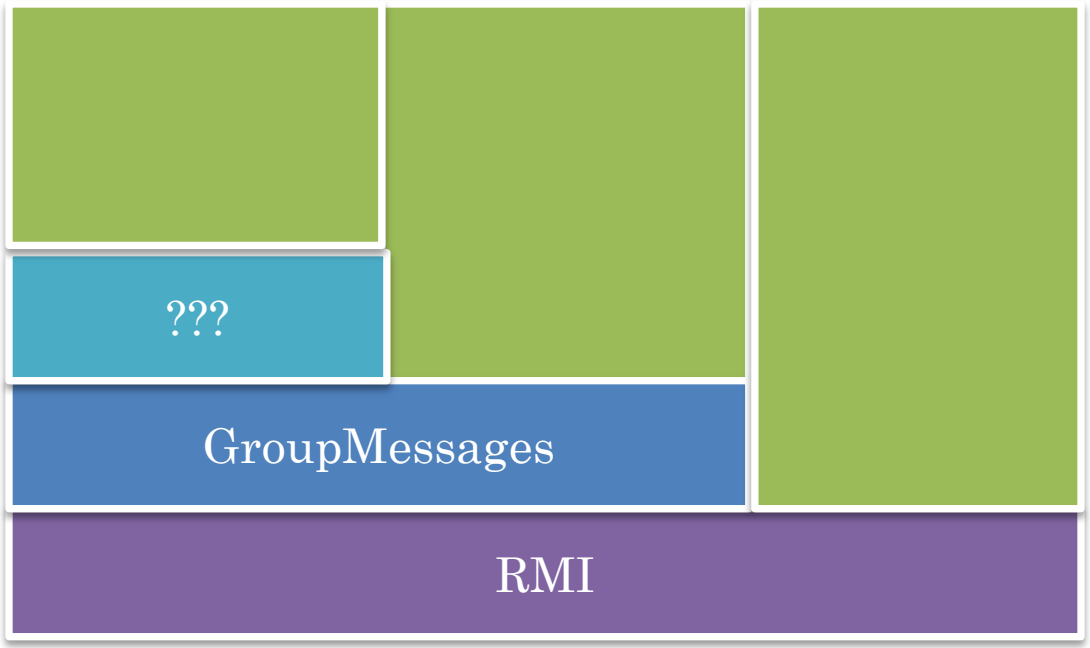


Group Messages Design



Dewan, P. (2014) *Programmer-Controlled Application-Level Multicast*. in *IEEE CollaborateCom*. IEEE.

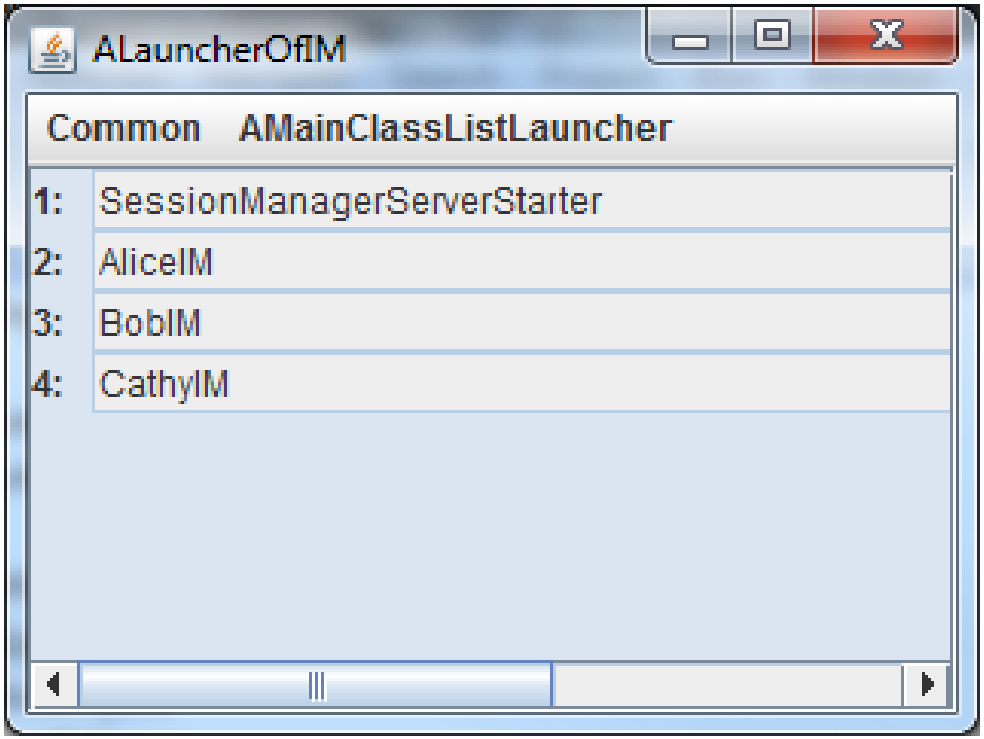
Group Messages Implementation



Easier to code as it is RPC, but synchronous, but feedthrough is an issue

Motivation for asynchronous RPC

Example (UI)


ALauncherOfIM
Common
AMainClassListLauncher
1:
SessionManagerServerStarter
2:
AliceIM
3:
BobIM
4:
CathyIM

Session Manager Starter

```
@Tags({DistributedTags.SERVER, DistributedTags.SESSION_MANAGER,
ApplicationTags.IM})
public class SessionManagerServerStarter {
  static ASessionManager server;
  public static void main (String[] args) {
   //do tracing
    …
   server = new ASessionManager();
   server.register(); //with RMI server
```

Generic Session Manager

Annotations provide typed comments like traces but are passive

Alice Starter

```
@Tags({DistributedTags.CLIENT_1, ApplicationTags.IM})
public class AliceIM implements ExampleIMSession{
public static final String USER_NAME = DistributedTags.CLIENT_1;
public static void main (String[] args) {
   String[] launcherArgs = {SESSION_SERVER_HOST, SESSION_NAME,
      USER_NAME,  APPLICATION_NAME,  Communicator.DIRECT};
  //do tracing
    …
   (new AnIMClientComposerAndLauncher()).
      oseAndLaunch(launcherArgs);
```

join(h^{SM}, s^1, u^2 , Editor, Direct)

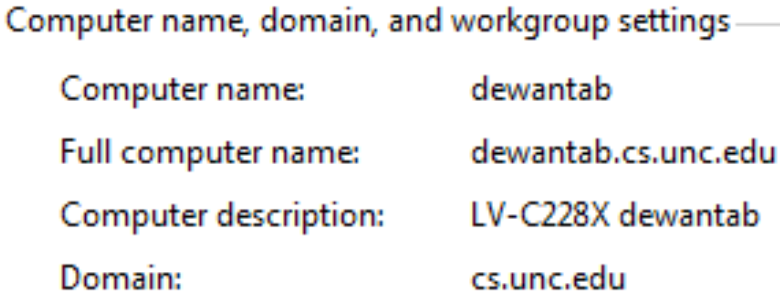
Generic Session Manager

Shared symmetric program

Host specification?

IM

Host Name




```
$ hostname
dewantab
```

```
public interface ExampleIMSession {
public static final String SESSION_NAME = "FrostySession";
public static final String APPLICATION_NAME = "IM";
public static final String SESSION_SERVER_HOST = "localhost";
}
```

localHost allows you to test same program on different hosts

Error Cause?

SessionManager

AliceIM [Java Ap

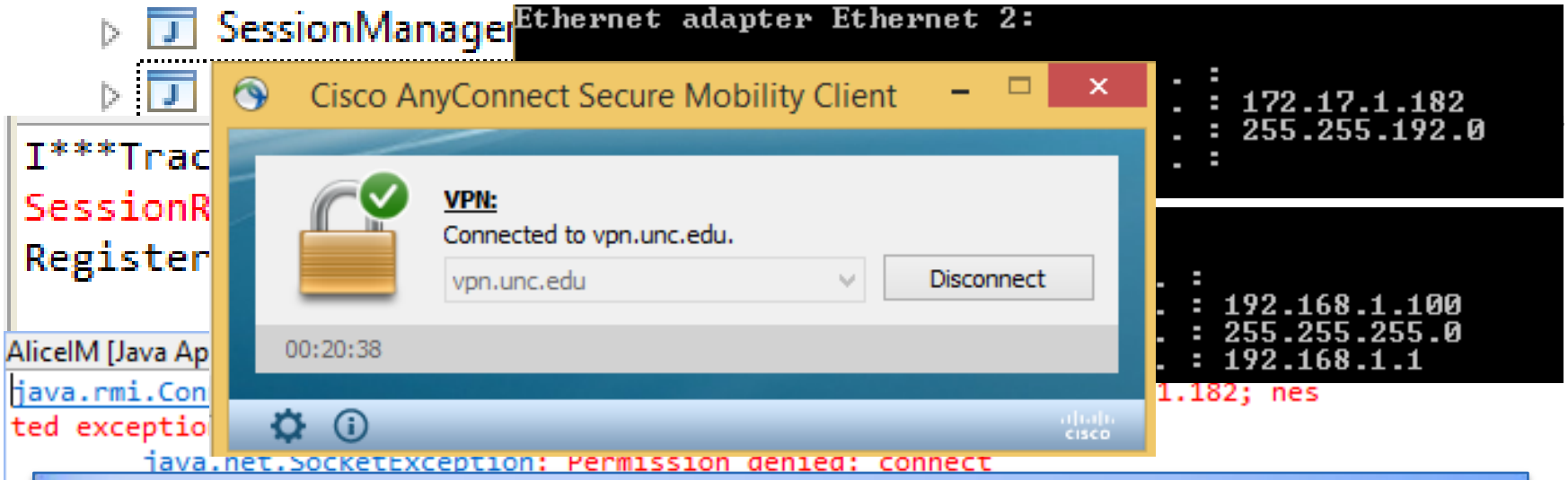
```
I***Tracer: showInfo
SessionRegistry: STAR
Registered Session Ma
```

```
Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 172.17.1.182
   Subnet Mask . . . . . . . . . . . : 255.255.192.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.1.100
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

AliceIM [Java Application] D:\Program Fil

```
java.rmi.ConnectIOException: Exception creating connection to: 172.17.1.182; nes
ted exception is:
        java.net.SocketException: Permission denied: connect
        at sun.rmi.transport.tcp.TCPEndpoint.newSocket(TCPEndpoint.java:631)
        at sun.rmi.transport.tcp.TCPChannel.createConnection(TCPChannel.java:216)
        at sun.rmi.transport.tcp.TCPChannel.newConnection(TCPChannel.java:202)
        at sun.rmi.server.UnicastRef.invoke(UnicastRef.java:129)
        at java.rmi.server.RemoteObjectInvocationHandler.invokeRemoteMethod(RemoteObjec
tInvocationHandler.java:194)
        at java.rmi.server.RemoteObjectInvocationHandler.invoke(RemoteObjectInvocationH
andler.java:148)
        at com.sun.proxy.$Proxy0.newMessage(Unknown Source)
        at util.session.AMessageSenderRunnable.run(AMessageSenderRunnable.java:71)
```

Error Cause?



The local host has a different meaning when you are using Cisco VPN!

Will get “access error” if Cisco VPN is connected

Giving host name directly does not help

Do not use Cisco VPM if using Cisco VPN!

Alice Starter

```
@Tags({DistributedTags.CLIENT_1, ApplicationTags.IM})
public class AliceIM implements ExampleIMSession{
public static final String USER_NAME = DistributedTags.CLIENT_1;
public static void main (String[] args) {
   String[] launcherArgs = {SESSION_SERVER_HOST, SESSION_NAME,
      USER_NAME,  APPLICATION_NAME,  Communicator.DIRECT};
  //do tracing
    …
   (new AnIMClientComposerAndLauncher()).
      oseAndLaunch(launcherArgs);
```

Generic Session Manager

Shared symmetric program

IM

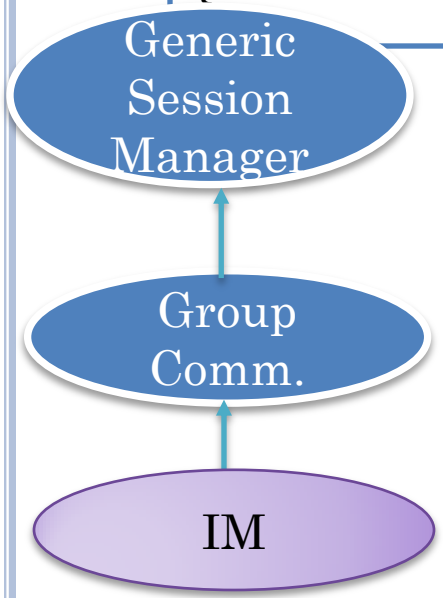
Join Call in Shared Program

```
public void compose(String[] args) {
   communicator = createCommunicator(args);
   super.compose(args);
   addCollaborationFunctions();
   doJoin();
}
…
protected void doJoin() {
    communicator.join();
}
```

Instantiate local group communication library

Add callbacks, before connecting to server

Invoke call on server



Should make createCommunicator as part of library

Multicast Arbitrary Serializable Objects

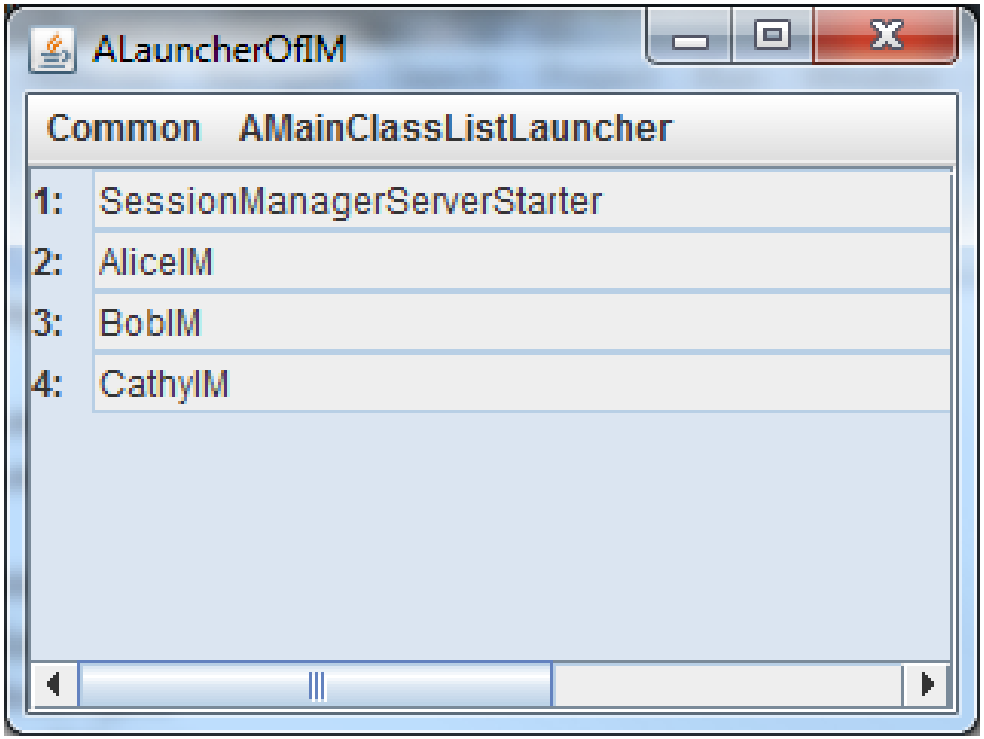
```
public synchronized void replicatedAdd(ElementType
anElement) {
  int anIndex = size();
  super.observableAdd(anIndex, anElement);
  if (communicator == null) return;
  ListEdit listEdit = new
     AListEdit<ElementType>(OperationName.ADD,
     anIndex, anElement, ApplicationTags.IM);
  communicator.toOthers(listEdit);
}
```

toOthers(msg)

```
public interface ListEdit<ElementType> extends Serializable {
  int getIndex();
  void setIndex(int anIndex);
  ElementType getElement();
  void setElement(ElementType
  …
}
```

RMI uses Add locking to projectlayer which requires the communicated objects to be labelled as Serializable, variables of only serializable superclasses can be communicated remotely

Example (UI) (Review)



Session Manager Starter (review)

```
@Tags({DistributedTags.SERVER, DistributedTags.SESSION_MANAGER,
ApplicationTags.IM})
public class SessionManagerServerStarter {
  static ASessionManager server;
  public static void main (String[] args) {
   //do tracing
    …
   server = new ASessionManager();
   server.register(); //with RMI server
```

Generic Session Manager

Alice Starter (Review)

```
@Tags({DistributedTags.CLIENT_1, ApplicationTags.IM})
public class AliceIM implements ExampleIMSession{
public static final String USER_NAME = DistributedTags.CLIENT_1;
public static void main (String[] args) {
   String[] launcherArgs = {SESSION_SERVER_HOST, SESSION_NAME,
      USER_NAME,  APPLICATION_NAME,  Communicator.DIRECT};
  //do tracing
    …
   (new AnIMClientComposerAndLauncher()).
      oseAndLaunch(launcherArgs);
}
```

Generic Session Manager

IM

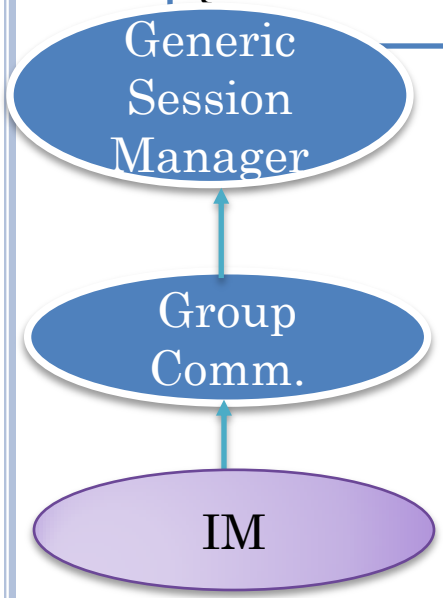
Join Call in Shared Program

```
public void compose(String[] args) {
   communicator = createCommunicator(args);
   super.compose(args);
   addCollaborationFunctions();
   doJoin();
}
…
protected void doJoin() {
    communicator.join();
}
```

Instantiate local group communication library

Add callbacks, before connecting to server

Invoke call on server



Should make createCommunicator as part of library

Multicast Arbitrary Serializable Objects

```
public synchronized void replicatedAdd(ElementType
anElement) {
  int anIndex = size();
  super.observableAdd(anIndex, anElement);
  if (communicator == null) return;
  ListEdit listEdit = new
     AListEdit<ElementType>(OperationName.ADD,
     anIndex, anElement, ApplicationTags.IM);
  communicator.toOthers(listEdit);
}
```

toOthers(msg)

Add operation marshalled into serializable object, reverse process at receiver

```
ementType> extends Serializable {
```

Programmer does marshalling and unmarshalling as multicast RPC does not exist and is hard to implement

RMI uses Add locking to projectlayer which requires the communicated objects to be labelled as Serializable, variables of only serializable superclasses can be communicated remotely

Receive Callback

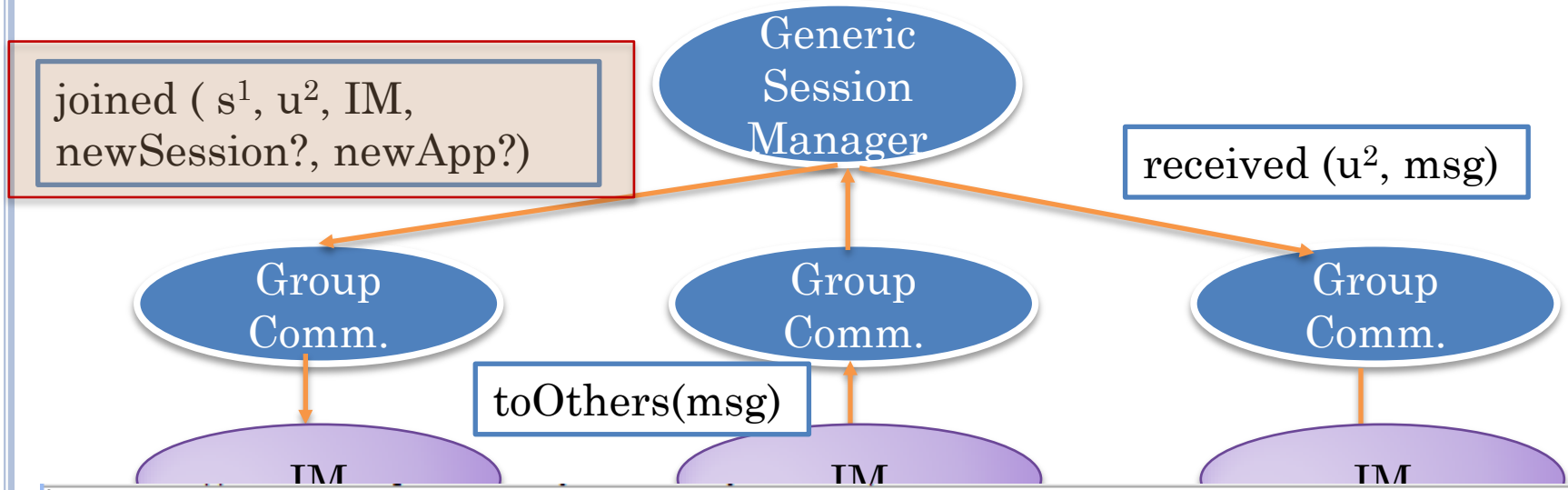
```
protected  void addHistoryInCoupler() {
  historyInCoupler = new AHistoryInCoupler(history);
  communicator.addPeerMessageListener(historyInCoupler);
}
```

```
public class AHistoryInCoupler implements PeerMessageListener {
 protected                                              ;
 public AH                                          Str
   history
  }
 public void objectReceived(Object message, String userName) {
   if (message instanceof ListEdit)
      processReceivedListEdit(
         (ListEdit<String>) message, userName);
 }
```

Can have multiple receive listeners processing different kinds of messages

Unmarshalling

Calls vs Callbacks

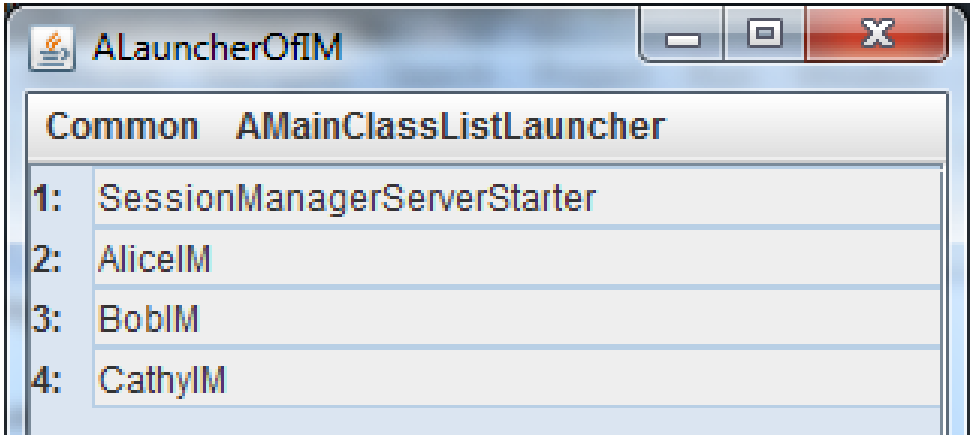



```
Please enter an input line or quit or history
The woods are lovely dark and deep
[Alice]The woods are lovely dark and deep
Please enter an input line or quit or history
[Bob]But I have promises to keep
[Cathy]And miles to go before I sleep
history
[Alice]The woods are lovely dark and deep, [Bob]But I have promises to keep, [Cathy]And miles to go before I sleep

Please enter an input line or quit or history
```

join(h^{SM}, s^1, u^2, IM, Relayed)

Join callback not needed in this application

Unaware Symmetric Join



Somehow all users know when to join

Could join when we know someone else has created and joined session

```
Please enter an input line or quit or history
The woods are lovely dark and deep
[Alice]The woods are lovely dark and deep
Please enter an input line or quit or history
[Bob]But I have promises to keep
[Cathy]And miles to go before I sleep
history
[Alice]The woods are lovely dark and deep, [Bob]But I have promises to keep, [Cathy]And miles to go before I sleep

Please enter an input line or quit or history
```

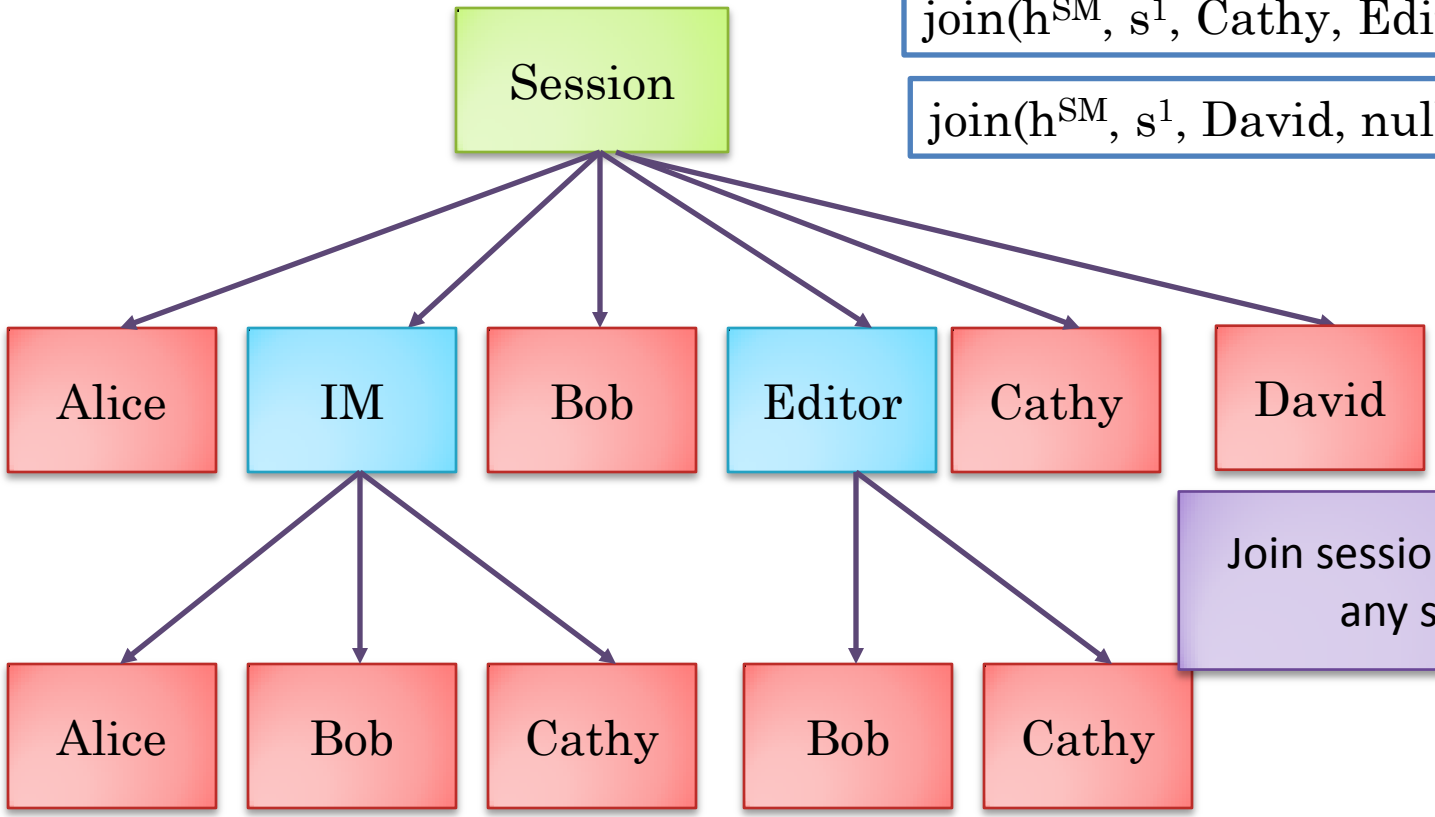
No awareness of out of band activities

Session with Application Sub-Sessions

join(h^{SM}, s^1, Cathy, IM, Relayed)

join(h^{SM}, s^1, Cathy, Editor, Relayed)

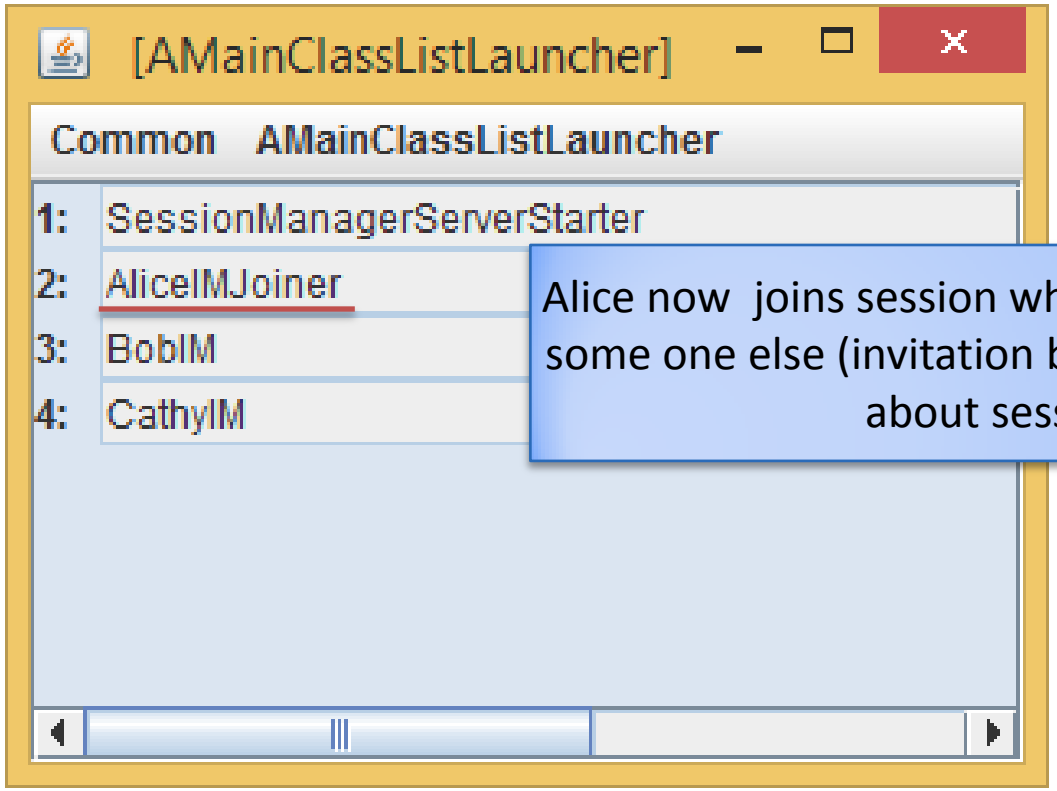
join(h^{SM}, s^1, David, null, Relayed)



Join session without joining any subsession?

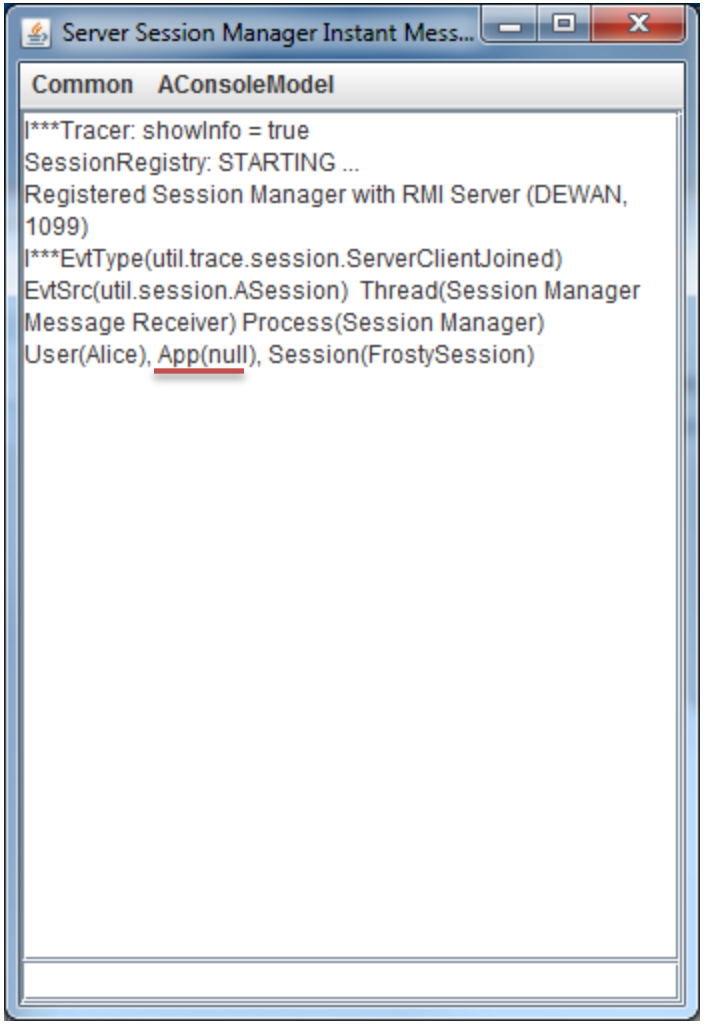
David's process cannot send or receive messages and simply listens to session callbacks, which can inform its user of session activity and join application sub-sessions

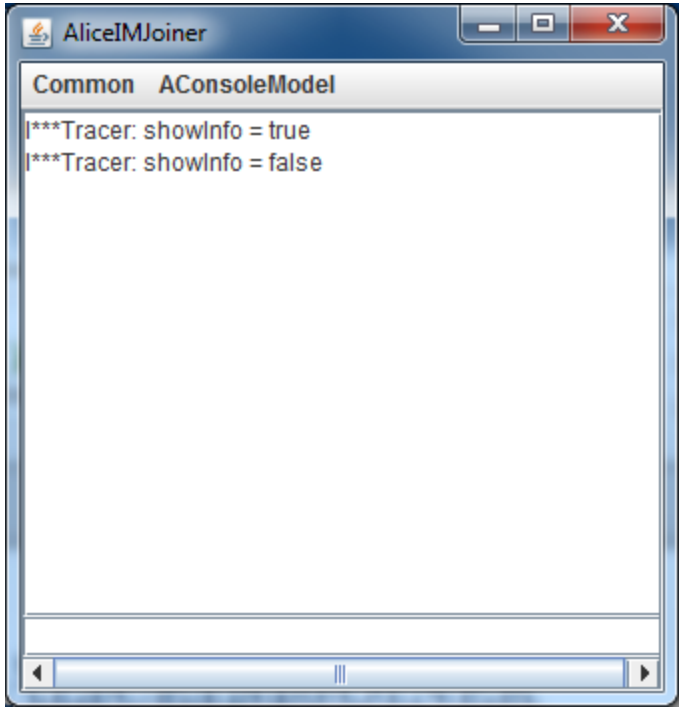
User Awareness



Alice now joins session when after it has been joined by some one else (invitation based joining) and is informed about session activities

Server and AliceIMJoiner

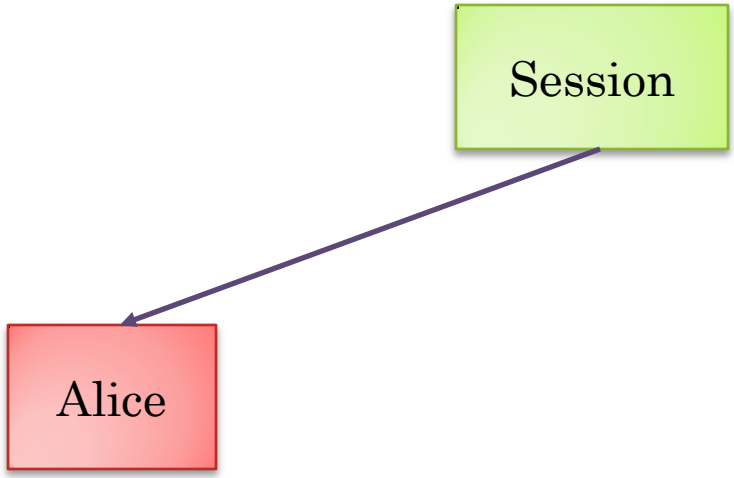




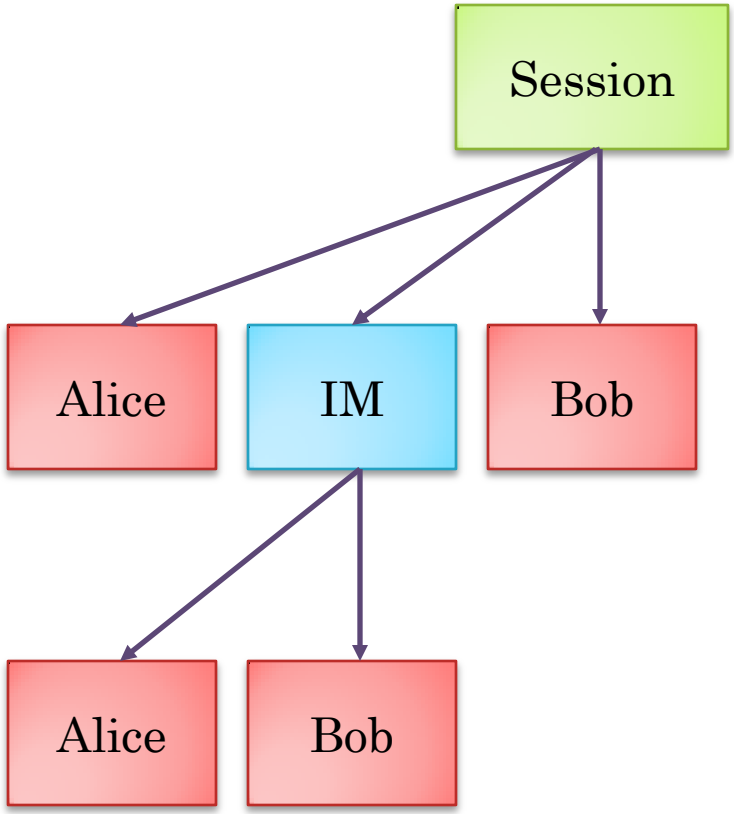
Session aware Joiner UI

No prompt, this is not the IM user interface

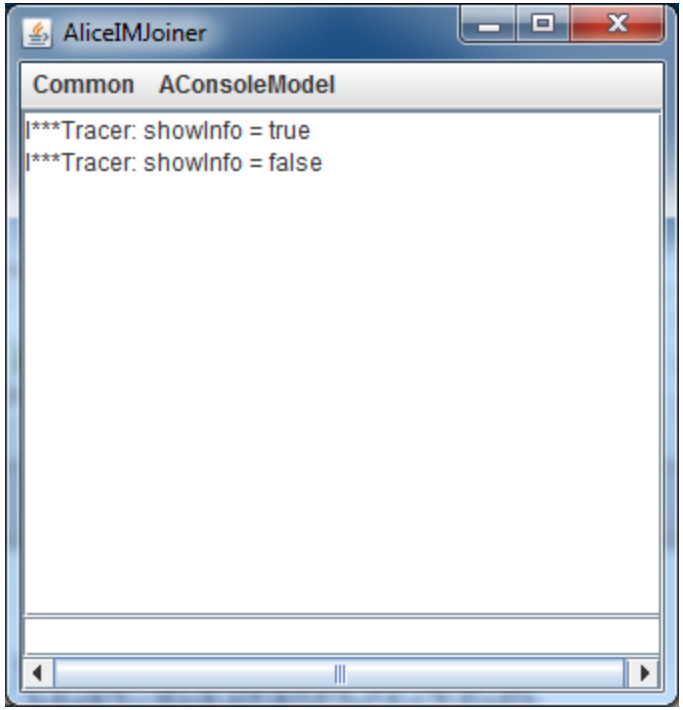
Session with Application Sub-Sessions



Session with Application Sub-Sessions

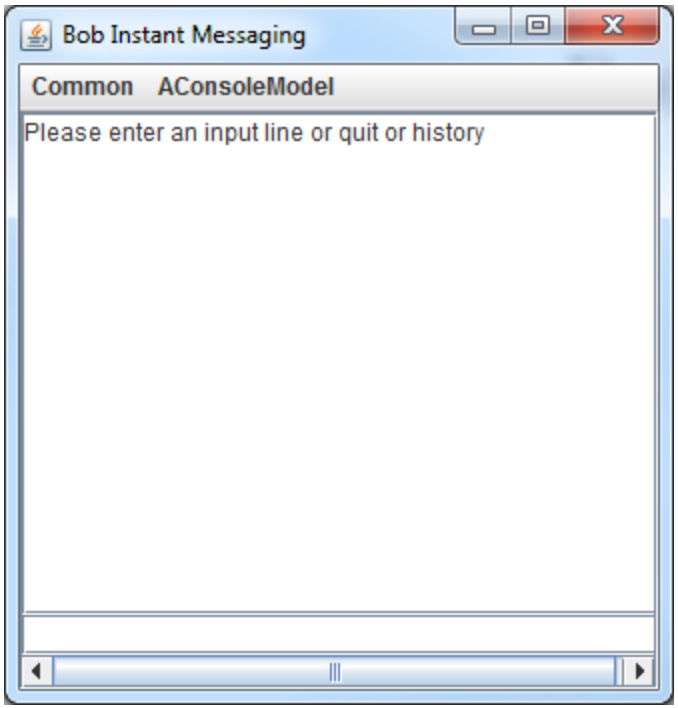


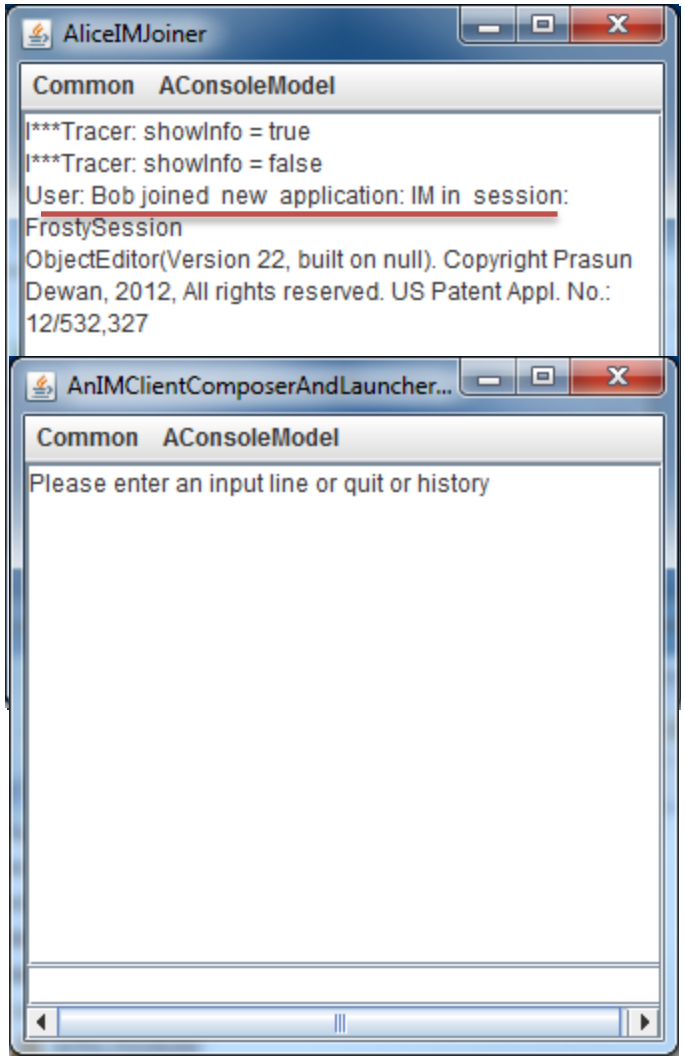
Server and AliceIMJoiner



Session aware Joiner UI

Bob Joins

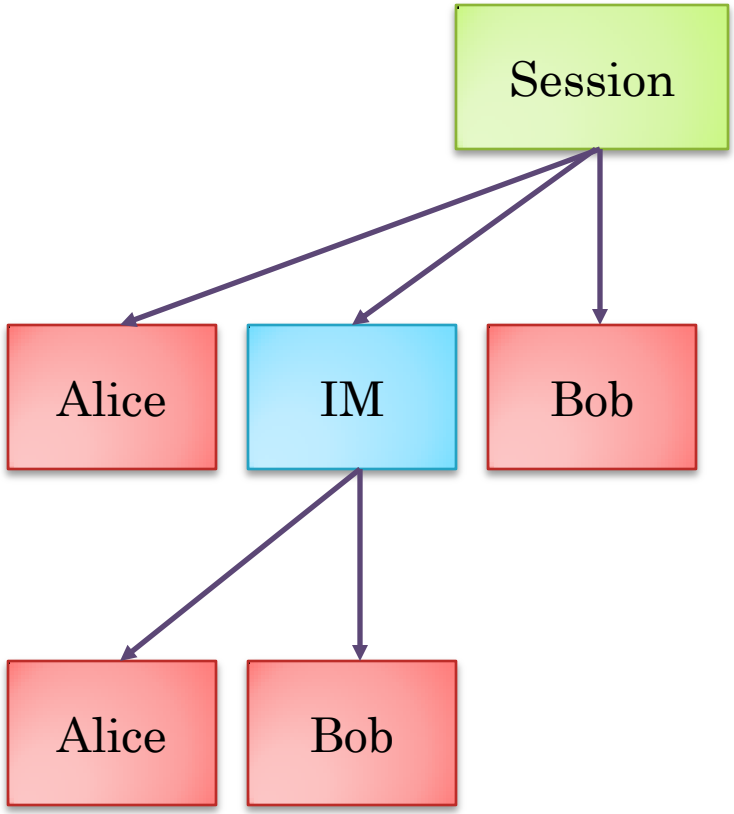


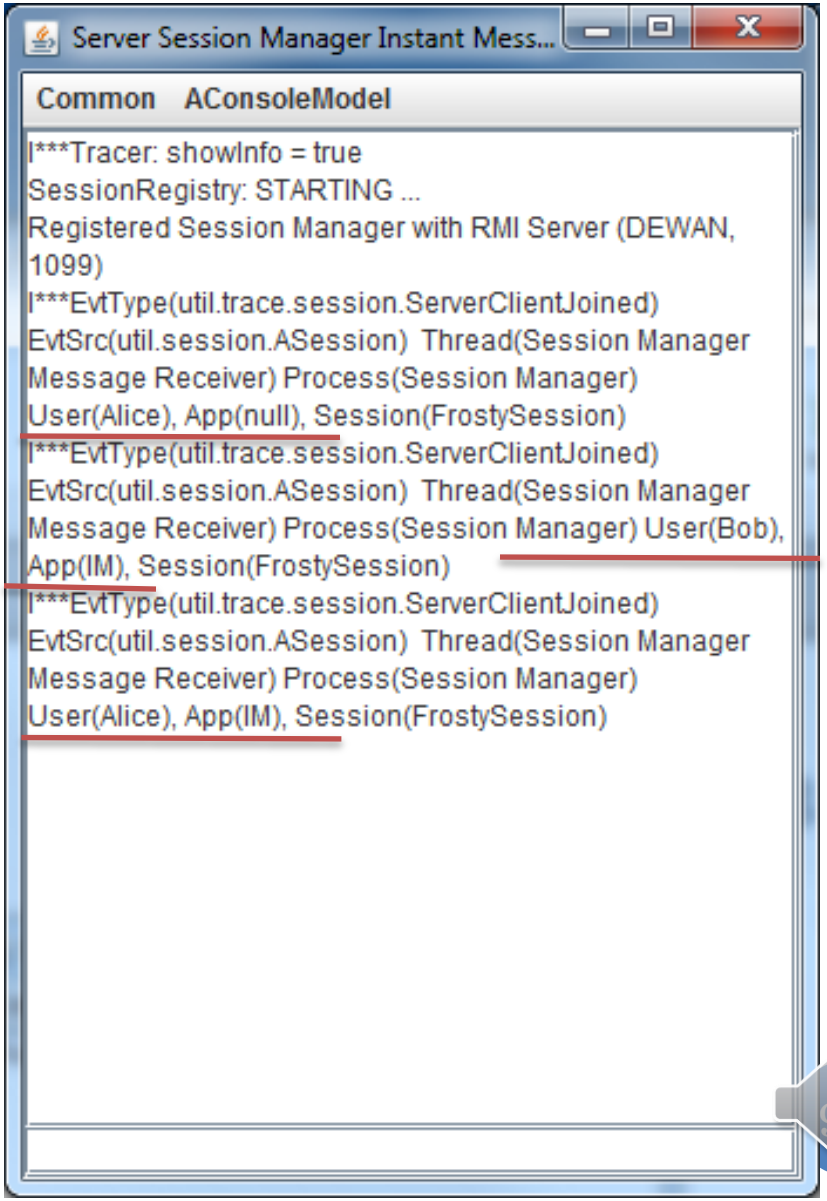


Session aware Joiner UI

IM UI

Session Manager Trace





AliceIMJoiner

```
public class AliceIMJoiner extends AliceIM{
public static void main (String[] args) {
  String[] launcherArgs = {SESSION_SERVER_HOST,
    SESSION_NAME, USER_NAME, null,  Communicator.DIRECT};
  (new AJoiningIMComposerAndLauncher()).compose(launcherArgs);
}
}
```

```
public  void addAwareness() {
  sessionJoiner = new ADynamicSessionJoiner
      (sessionServerHost, userName);
  communicator.addSessionMessageListener(sessionJoiner);
}
```

A different program is run by Alice's session aware joiner

Receive Callback and Forking JVM

```
public class ADynamicSessionJoiner implements
SessionMessageListener {
   …
public void clientJoined(
String aUserName, String anApplicationName, String aSessionName,
  boolean aNewSession, boolean aNewApplication, Collection<String>
anAllUsers) {
   printAwarenessMessage(aUserName,  anApplicationName,
    aSessionName, aNewSession,  aNewApplication, anAllUsers);
   if (aNewApplication && anApplicationName != null &&
       DEFAULT_APPLICATION_NAME.equals(anApplicationName))
    joinSession(anApplicationName, aSessionName);
  }
}
public void joinSession(String
aSessionName) {
  String[] launcherArgs = {sessionManagerHost, aSessionName,
     userName, anApplicationName, Communicator.DIRECT};
  OEMisc.runWithObjectEditorConsole
      (AnIMClientComposerAndLauncher.class, launcherArgs);
}
```

The original program in previous version

Session Manager/Communicator Steps

- ClientJoinFinished.java
- ClientJoinInformationUpdated.java
- ClientJoinInitiated.java
- ClientJoinNotificationDistributedToListeners.jav
- ClientJoinNotificationMarshalled.java
- ClientJoinNotificationReceived.java
- ClientJoinNotificationUnmarshalled.java
- ClientLeaveInformationUpdated.java
- ClientLeaveNotificationDistributedToListeners.j
- ClientLeaveNotificationMarshalled.java
- ClientLeaveNotificationReceived.java
- ClientLeaveNotificationUnmarshalled.java
- ClientReceivedObjectUnmarshalled.java
- DataReceiveMarshalled.java
- DelayedMessageInfo.java
- DelayVariationSet.java
- JoinRequest.java
- JoinRequestMarshalled.java
- LeaveRequest.java
- LeaveRequestMarshalled.java
- MessageBuffered.java
- MessageBufferInfo.java
- MessageBufferReferenceCountDecremented.java
- MessageCopied.java
- MessageForwarded.java
- MessageGivenToFilter.java
- MessageInfo.java
- MessagePutInQueue.java
- MessageReceived.java
- MessageRetrievedFromQueue.java
- MessageSent.java
- MessageUnBuffered.java
- MinimumDelaySet.java
- MulticastGroupCreated.java
- MulticastGroupJoinInformationUpdated.java
- MulticastGroupLeaveInformationUpdated.java

Not all important steps traced and at least one step deprecated

Session Manager/Communicator Steps (Contd)

- ReceivedMessageDelayed.java
- ReceivedMessageDistributedToListeners.java
- SendDataRequest.java
- SentMessageDelayed.java
- ServerClientJoined.java
- ServerClientLeft.java
- ServerJoinNotificationDeliveredToListeners.java
- ServerJoinRequestMarshalled.java
- ServerJoinRequestUnmarshalled.java
- ServerLeaveNotificationDeliveredToLocalObserver
- ServerLeaveRequestMarshalled.java
- ServerLeaveRequestUnmarshalled.java
- ServerRemoteJoinNotificationMarshalled.java
- ServerRemoteJoinNotificationSent.java
- ServerRemoteLeaveNotificationMarshalled.java
- ServerRemoteLeaveNotificationSent.java
- SessionCreated.java
- SessionInfo.java
- SessionJoinInformationUpdated.java
- SessionLeaveInformationUpdated.java
- SessionMessageReceived.java

- ThreadCreated.java
- ToAllDateSendMarshalled.java
- ToOthersDataSendMarshalled.java
- ToUserDataSendMarshalled.java

Summary

- Distributed architecture = process + object architecture
- General and special distributed architectures exists, which depend on distributed communication layer
- At this point, looking at general architecture
- IPC provides the most general architecture but maybe too general, not providing support for
- With IPC need to build
  - own session manager for dynamic sessions, group multicast, choice between relayed and direct communication, threading
- Can build group communication automating this
- Session vs. application-session
- Relayed vs direct communication
- All, specific user, all multicast groups
- Synchronous vs. asynchronous
- Threads to balance latency vs. consistency
- Symmetric vs asymmetric join
- Serialization and marshalling
- Session awareness

Next

- How to use group communication for different classes of applications
  - Model-based sharing
  - Window-based sharing
- How do build higher level abstractions for these classes?